Table of Contents

Table of Contents (Cont'd.)

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (Cont'd.)

Figure

LIST OF TABLES

NASA Contractor Report 2932

Digital Computer Processing of LANDSAT Data for North Alabama

A. D. Bond, R. J. Atkinson,
M. Lybanon, and H. K. Ramapriyan

CONTRACT NAS8-21805
DECEMBER 1977

NASA

NASA Contractor Report 2932

Digital Computer Processing of LANDSAT Data for North Alabama

A. D. Bond, R. J. Atkinson,
M. Lybanon, and H. K. Ramapriyan
Computer Sciences Corporation
Huntsville, Alabama

Prepared for
George C. Marshall Space Flight Center
under Contract NAS8-21805

NASA
National Aeronautics
and Space Administration

Scientific and Technical
Information Office

1977

FOREWORD

The rationale for the division into chapters in this report was to describe the processing of Landsat data from various viewpoints. Chapter I is a description of the Landsat system, and the origin, type, and handling of its generated data. Chapter II is a verbal description of the analysis procedures; Chapter III is a mathematical descriptic . of the analysis techniques. Chapter IV is a presentation of the results achieved for Landsat coverage of North Alabama, while Chapter V is documentation of the major computer programs used in the analysis.

I. THE LANDSAT SYSTEM

1-1. INTRODUCTION

Landsat-1 (formerly ERTS-1) is an experimental satellite whose purpose is to investigate the feasibility of remote sensing from space as a practical approach to efficient management of the earth's resources. The principal disciplines involved are agriculture, forestry, geology, geography, hydrology, and oceanography.

For this purpose, the satellite acquires repetitive multispectral images of the earth's surface and transmits this raw data through ground stations to a data processing center at the NASA Goddard Space Flight Center, for conversion into black and white or color photographs and computer tapes to fulfill the varied requirements of investigators and user agencies. Thus, such a remote sensing vehicle, due to its ability to cover large areas and generate large amounts of data, becomes a major element in an earth survey system, such as that illustrated in Figure 1. Several elements of this system will be discussed in the present report.

1-2. DESCRIPTION OF THE LANDSAT SYSTEM

Landsat-1 was launched in July, 1972 and traverses a circular, sun synchronous, near-polar orbit at an altitude of 915 km. (570 miles). This orbit provides repetitive earth coverage under nearly constant observation conditions, i.e. solar times. The satellite circles the earth every 103 minutes, completing 14 orbits per day, and views the entire earth every 18 days. Orbit specifications require that the satellite ground trace repeat its earth coverage at the same local time every 18 day period within 37 km. (23 miles). A typical one-day ground coverage trace is shown in Figure 2 for the daylight portion of each orbital revolution.

The overall Landsat system is illustrated in Figure 3. The satellite carries a payload of imaging multispectral sensors, wideband video tape recorders, and the spaceborne portion of a Data Collection System. The video data is received at Fairbanks, Alaska, Goldstone, California, and the Network Test and Training Facility (NTTF) at Goddard Space Flight Center (GSFC). Video data stored on magnetic tapes is received by the NASA Data Processing Facility (NDPF). The NDPF then performs the video-to-film conversion and correction, producing black and white images from individual spectral bands and color composites from several spectral bands. The NDPF includes a storage and retrieval system for delivery of data products and services to the investigators and other data users.

The satellite system consists of the earth resources payload subsystem and the various support subsystems comprising the spacecraft vehicle. The configuration is shown in Figure 4. Control of observatory attitude to the local

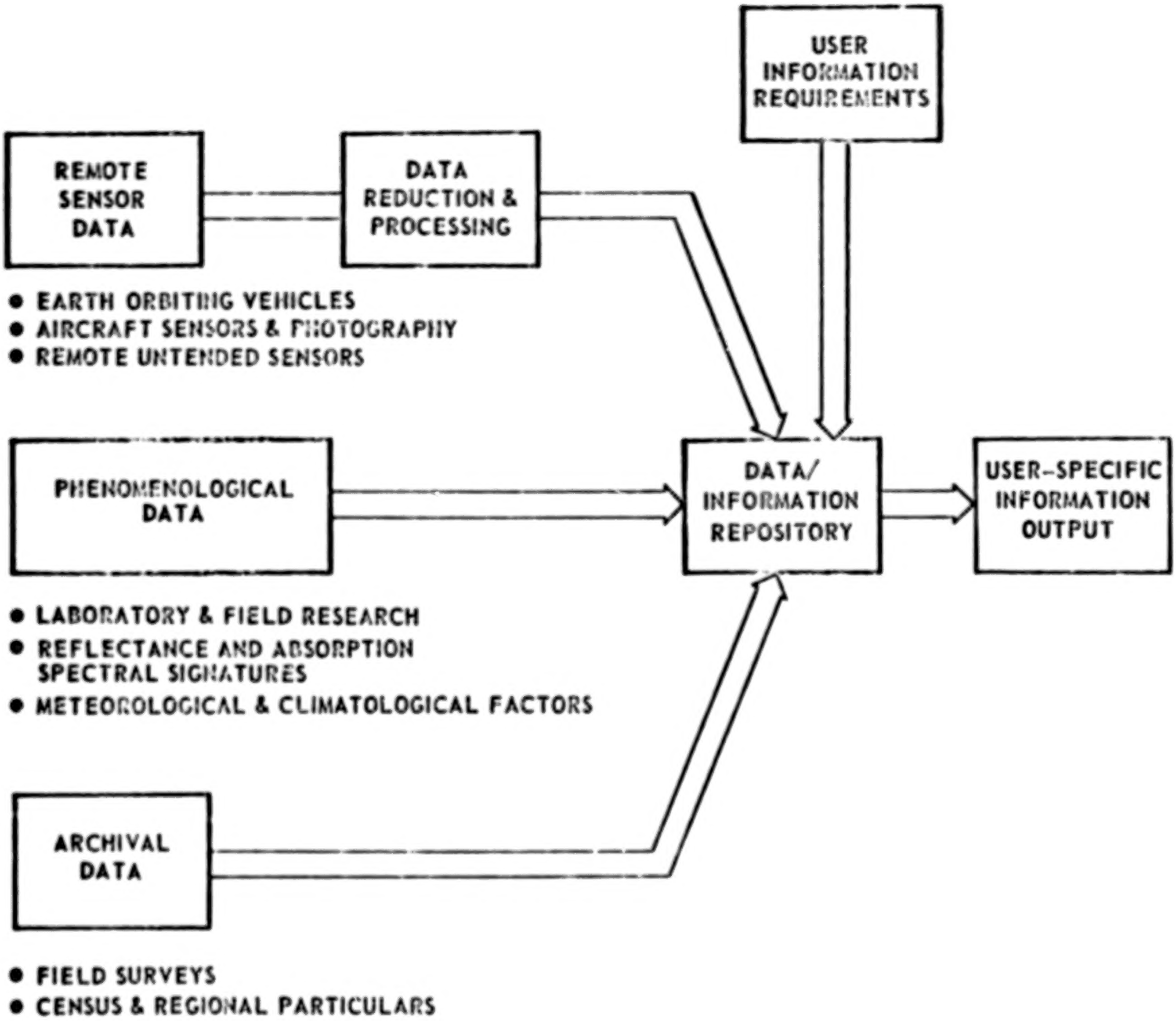


Figure 1. Basic Elements of an Earth Survey Information System.

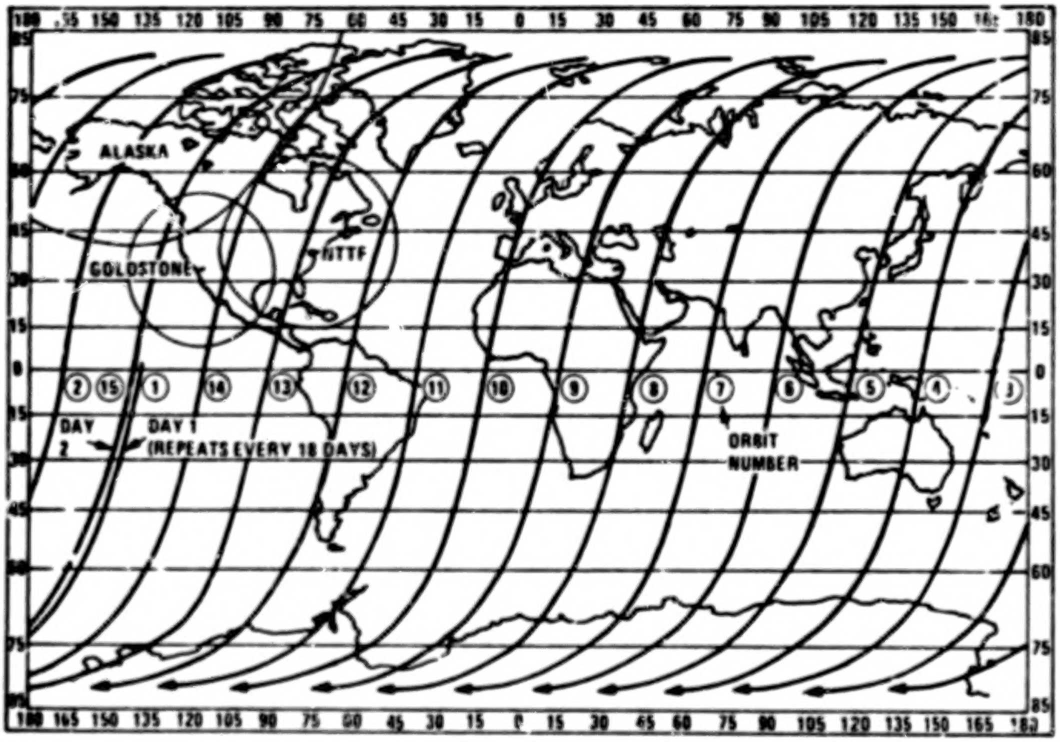


Figure 2. Landsat Ground Coverage.

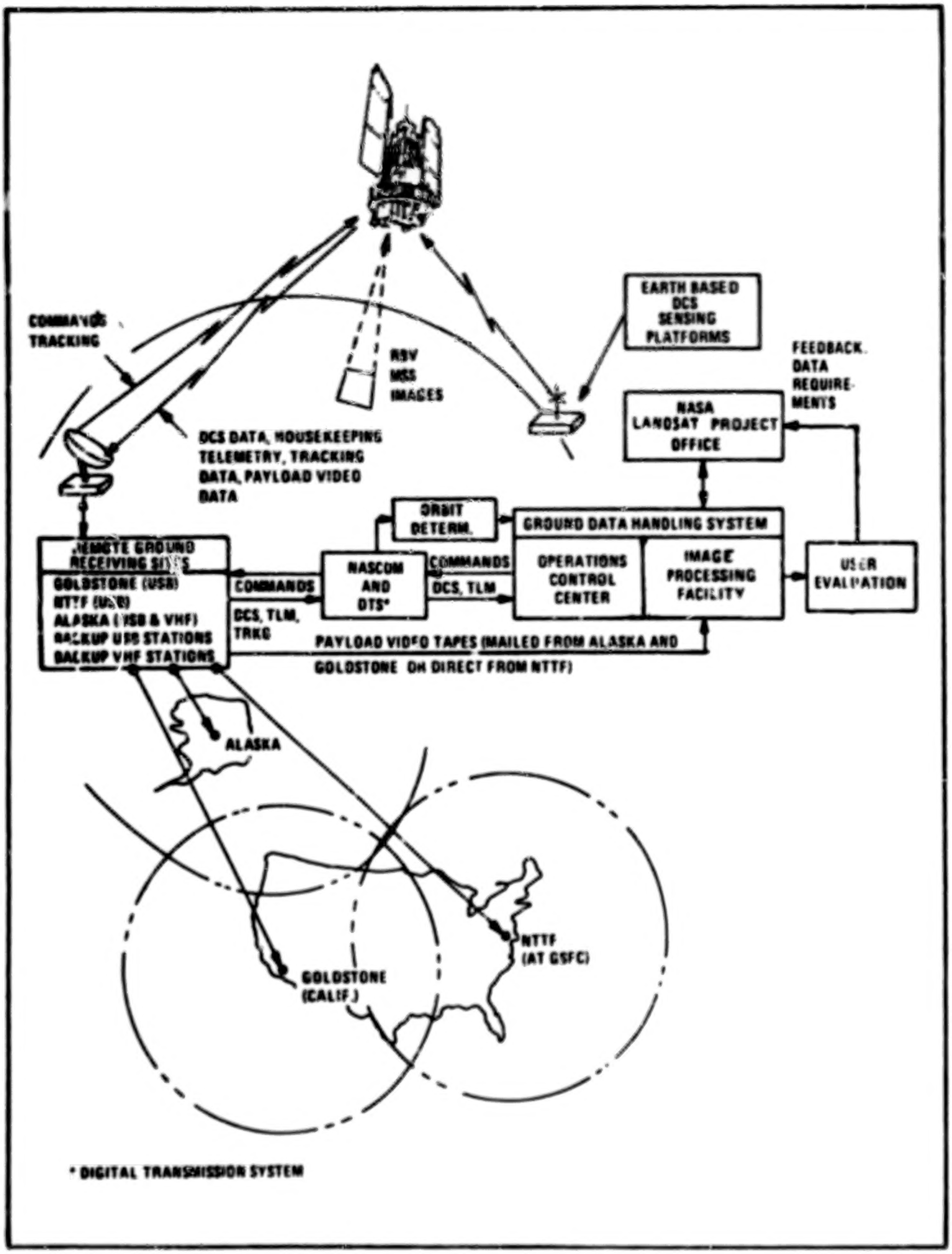


Figure 3. Landsat Support System

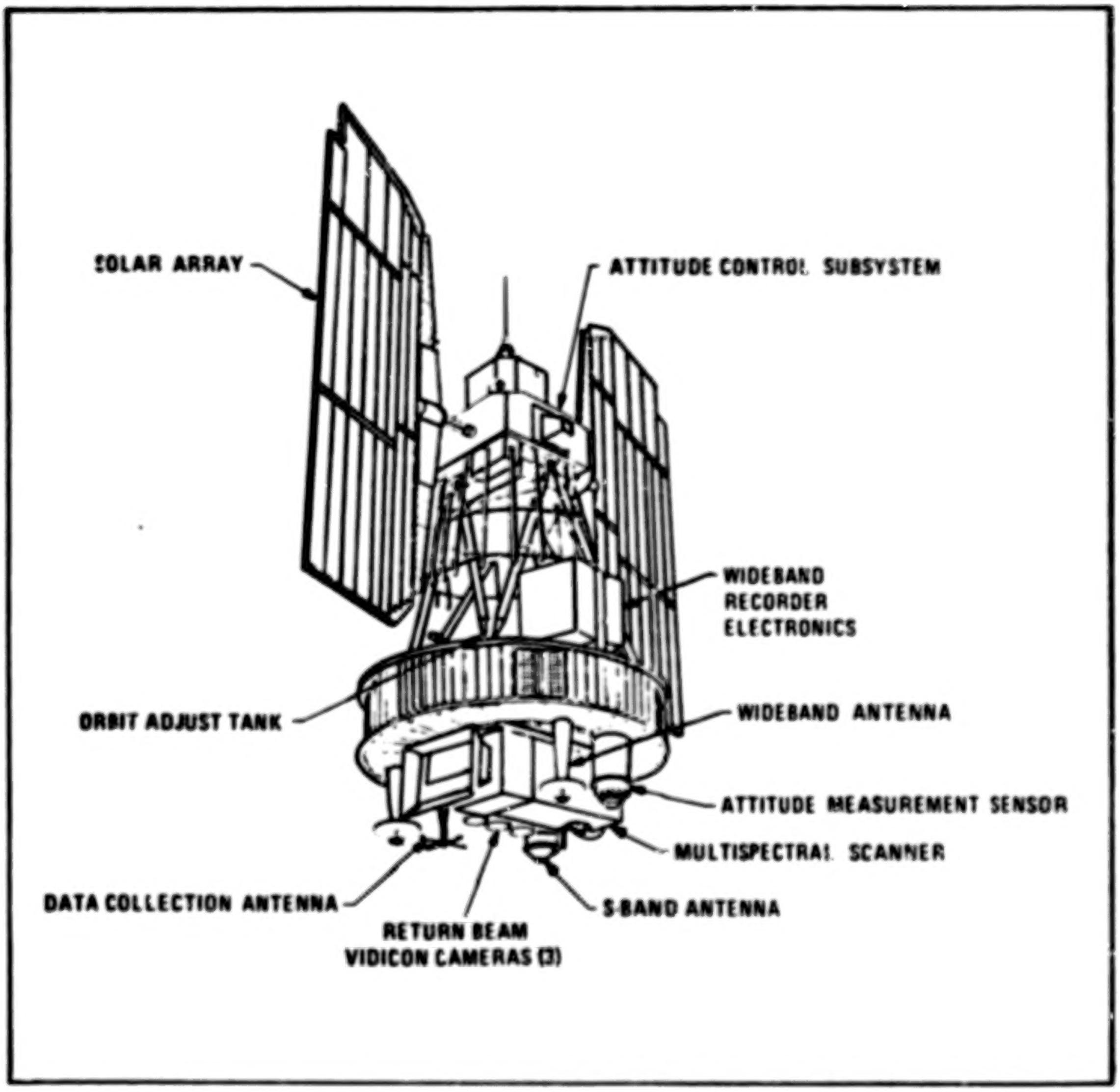


Figure 4. Landsat Satellite Configuration

vertical and orbit velocity vectors within 0.7 degree of each axis is achieved by a three axis active Attitude Control Subsystem. An independent passive Attitude Measurement Sensor provides pitch and roll attitude data accurate to within 0.07 degree to aid in image location. Orbit adjustment capability is provided by a system of one-pound thrusters. Payload video data are transmitted to ground stations over two wideband S-Band data links. Electrical power is generated by two solar arrays, with storage provided by batteries for spacecraft eclipse periods.

The earth resources payloads are the return beam vidicon (RBV) camera system and the multispectral scanner (MSS). The return beam vidicon camera operates by shuttering three independent cameras simultaneously, each sensing a different spectral band in the range 0.48 to 0.83 micrometers. The viewed ground scene, 185 x 185 km. (115 x 115 miles) in size, is stored on the photosensitive surface of the camera tube and, after shuttering, the image is scanned by an electron beam to produce a video signal output.

The multispectral scanner is a scanning device which uses an oscillating mirror to scan over lines perpendicular to the spacecraft ground track as shown in Figure 5. The surface of the earth is imaged in four spectral bands through the same optical system, so that optical energy is sensed simultaneously in the four bands. The bands lie in the solar reflected spectral region, and their wavelength limits are as follows:

Band 1	0.5 to 0.6 micrometers
Band 2	0.6 to 0.7 micrometers
Band 3	0.7 to 0.8 micrometers
Band 4	0.8 to 1.1 micrometers

Bands 1 through 3 use photomultiplier tubes as detectors; Band 4 uses silicon photodiodes. The cross track ground coverage of 185 km. (115 miles) is obtained as the flat mirror oscillates $\pm$ 2.89 degrees at a rate of 13.62 Hz. As the image is thus swept across an array of optical fibers, light impinging on each glass fiber is conducted to an individual detector through an optical filter, unique to the appropriate spectral band. The detector outputs are sampled, digitized, and formatted into a continuous data stream of 15 megabits per second. The along-track scan is produced by the orbital velocity of the satellite, which causes an along-track motion of the subsatellite point of 6.47 km/sec. (4.0 miles/sec). The mirror oscillation frequency is such that the subsatellite point traverses 474 meters during the 73.42 millisecond scan and retrace cycle. During each mirror cycle, six lines of 79 meters width are scanned, and hence the line scanned by the first detector in a cycle is adjacent to the sixth line of the previous cycle. This scan pattern is shown in Figure 6. The instantaneous field of view of 79 meters (86.4 yards) square on the ground is delineated by the square input end of each optical fiber. The area sampled to form the reflectance data for each picture element (pixel) is 6241 square meters (1.54 acres). Along

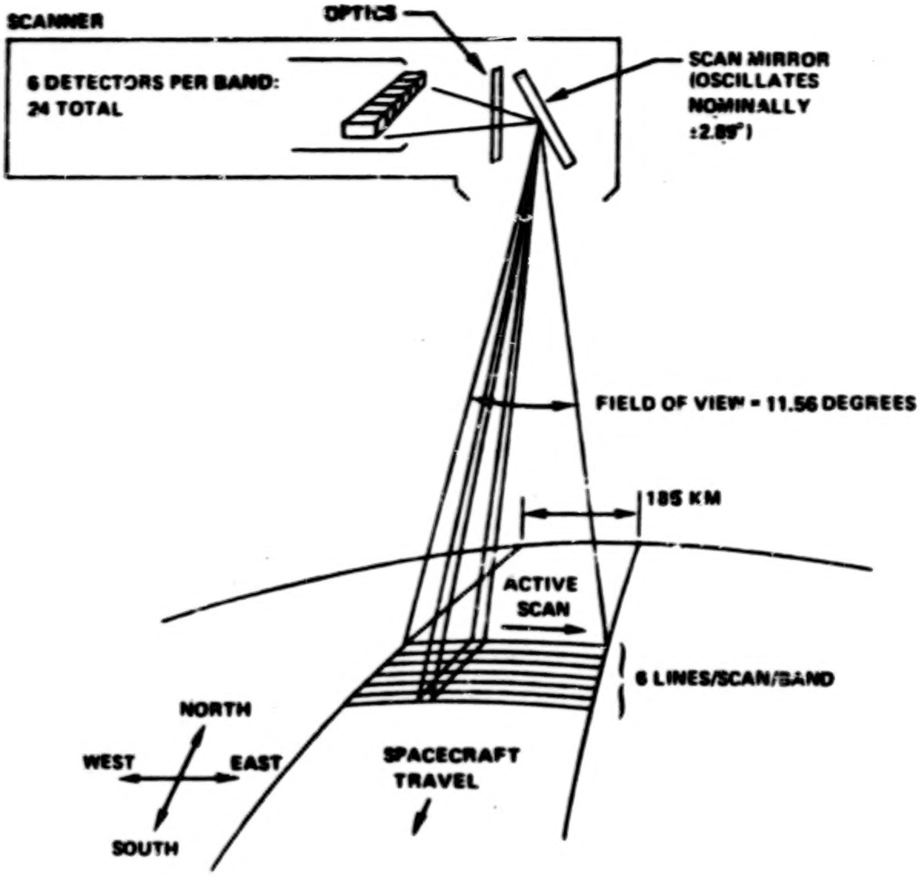


Figure 5. MSS Scanning Arrangement

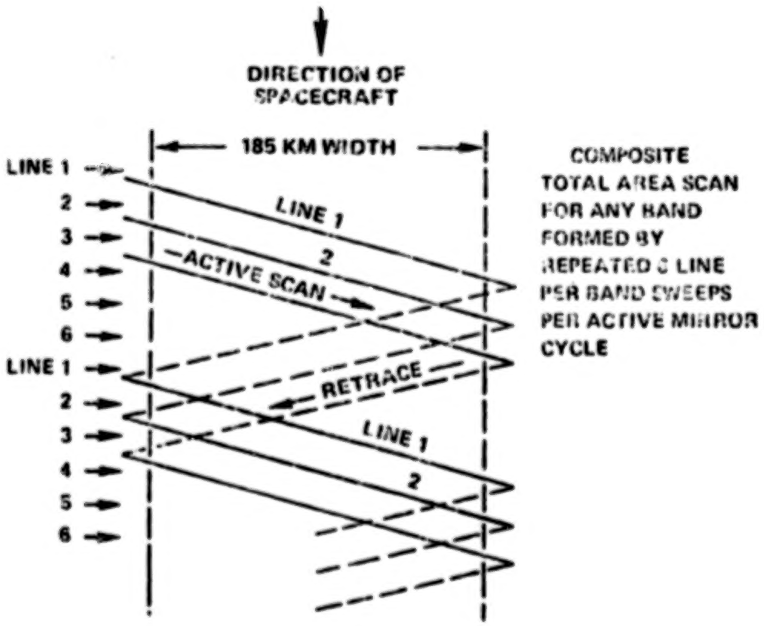


Figure 6. Ground Scan Pattern of the MSS.

a scan line the sampling rate is approximately 100,000 Hz, which results in overlap of the samples along the scan lines such that the effective area covered per sample is 1.1 acres.

1-3. ORIGIN AND TYPE OF DATA

The physical origin of the electromagnetic energy that is detected quantitatively in the Landsat sensors is reflection of sunlight from the target scenes on the earth's surface. The ability to classify the measurements according to their origin from various objects arises from variations in the reflection as a function of wavelength. The physiological equivalent of this process is the recognizing of an object by its color only.

The quantitative definition of spectral composition is often called the spectral 'signature', and this represents the distribution of intensity of radiation as a function of wavelength. Each category, species or material will in general have a unique distribution, and it is this distribution, or signature, that is used for identifying the species. Over the visible range of the spectrum, the spectral signature may be thought of as the color distribution of the species in question.

Spectral signatures may be illustrated by plots of radiance intensity versus wavelength, ideally as is shown in Figure 7. In order to apply data analysis algorithms, it is necessary to represent these curves in numerical form, but a numerical point by point plot is not used in order to avoid data proliferation. Rather, a set of wavebands is selected over which the variation of spectral radiance is sufficiently large to permit discrimination between curves. Each spectral signature can then be represented as a set of numerical values that provide measures of the predominant spectral components present. In many multispectral sensor systems, the chosen separation of the wavebands is determined by the spectral resolution of the detector arrays or interference filter.

For analysis purposes, it is convenient to consider the set of numerical values derived from a single signature spectrum as a vector, and to represent this vector in an orthogonal vector-space whose axes are identified with the spectral components of the signature. This form of representation is illustrated in Figure 8.

The vector space interpretation of spectral signatures is particularly convenient for automatic data analysis since the algorithms of pattern recognition and feature classification can be applied directly. Each signature may be considered as characteristic of a certain class, species, or category. The vector components of any signature may be considered as characteristic features that enable the associated class to be distinguished from other, perhaps related, classes. As long as known characteristic signatures are available for comparison, feature vector measurements derived from remote sensor data may be sorted or classified by automatic decision logic according to the species from which they originated.

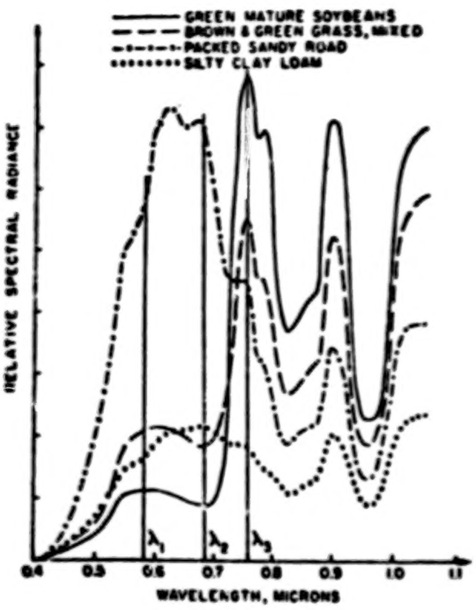


Figure 7. Relative Spectral Radiance Signatures of Agriculture Scenes

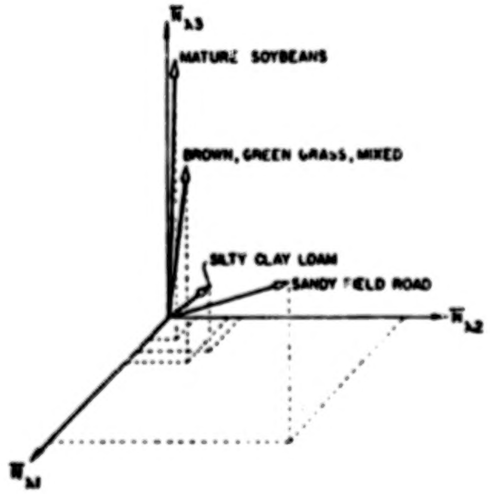


Figure 8. Spectral Signatures Interpreted as Feature Vectors

The format of the data output from the multi-channel detector array is particularly convenient for computer data processing. The signal output of each detector is recorded on a magnetic tape, and the variation of the recorded signal along the length of the magnetic tape then bears a very close correlation to the physical content of the ground scene, and each channel recorded on the tape may be regarded as the record of the ground scene viewed in a different color band, as depicted in Figure 9.

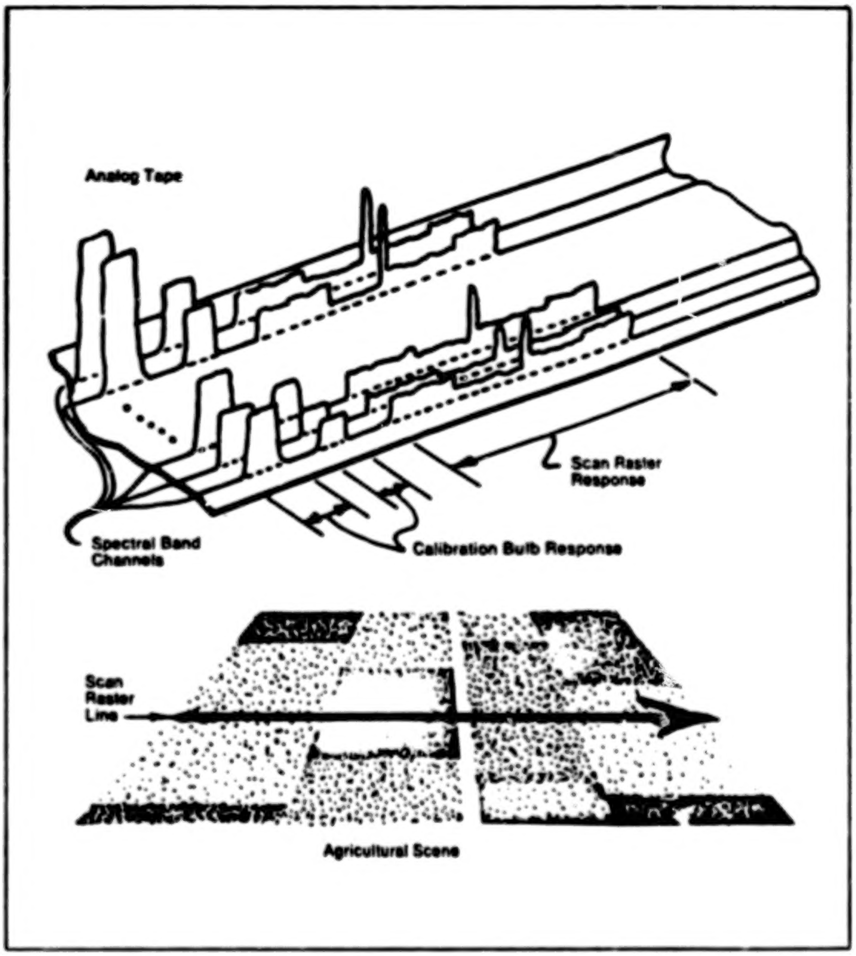
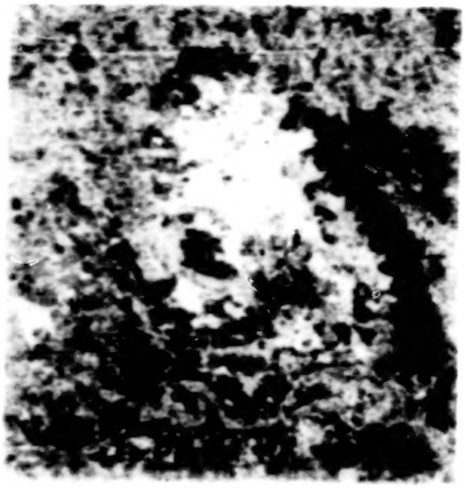


Figure 9. Correlation of Tape Record of Detector Outputs with Physical Features of Ground Scene

The four spectral channels employed in the Landsat multispectral scanner lie approximately in the green, red, near infrared, and infrared ranges. Figure 10 shows data from the four bands in a 500 x 500 pixel area including Huntsville, Alabama, acquired on November 4, 1972.

1-4. DATA HANDLING AND DATA PRODUCTS

The Landsat data in digital format is segmented into four computer compatible tapes (CCTs). Each tape contains identification and annotation records,



Green Band

Red Band

Near Infrared Band

Infrared Band

Figure 10. Landsat Data in Four Bands Covering Huntsville, Alabama.

followed by the data in eight-bit bytes. The annotation record contains information regarding the conditions of exposure, such as date and time, sun elevation, coordinates of the image center, locations of lines of latitude and longitude intersecting the image. Data values in the four channels from pairs of adjacent pixels are interleaved according to the following order of channel numbers:

1 1 2 2 3 3 4 4.

Thus, the first step in data handling is the idetnification of the data segment required, based on latitude and longitude, possibly reording to place data values for each pixel in channel order, and conversion to integer or decimal type numbers.

It is then necessary to use some means of examining the data visually in terms of the density levels, that is, reconstruct the image of the ground scene. This is to verify that the scene of interest has indeed been selected, and to locate specific locations and land use types for input to certain processing steps.

One method of displaying digital data is by plotting on the computer line printer with certain data values being represented by specific characters. The darkness of the printing is increased by overprinting several characters at the same location. This method has the advantages of showing the values of the data and of allowing exact determination of the coordinates of each data value. Disadvantages are poor gray level rendition and the inability to display large regions. An example of a printer plot is given in Figure 11, showing the Landsat data of the Huntsville Jetport.

The Landsat scanner data is more easily interpreted when displayed on a cathode ray tube (CRT), which may be either a storage type or a TV monitor. In the present work, the former type, a Dicomed was used. This device is capable of displaying 64 gray levels, and the screen size is 2048 by 2048 pixels.

In order to obtain photographic prints of scanner data and land use maps, a film writer is used. This device reproduces scan lines read from magnetic tape on film, with the film density being proportional to the numbers read from tape. An Optronics model Photowrite was used, and the film writer, tape drive, and film scanner-digitizer are shown in Figure 12. Prints in this report, such as Figure 10, were produced by this method.

The line printer plots are a very useful output product since they can be obtained in the same computer run which has processed the data. A 500 by 500 pixel area, such as that shown in Figure 10, can be plotted in the width of the printout if every fourth pixel in each direction is plotted. This allows, for example, immediate examination of a land use map during the process of computer classification.

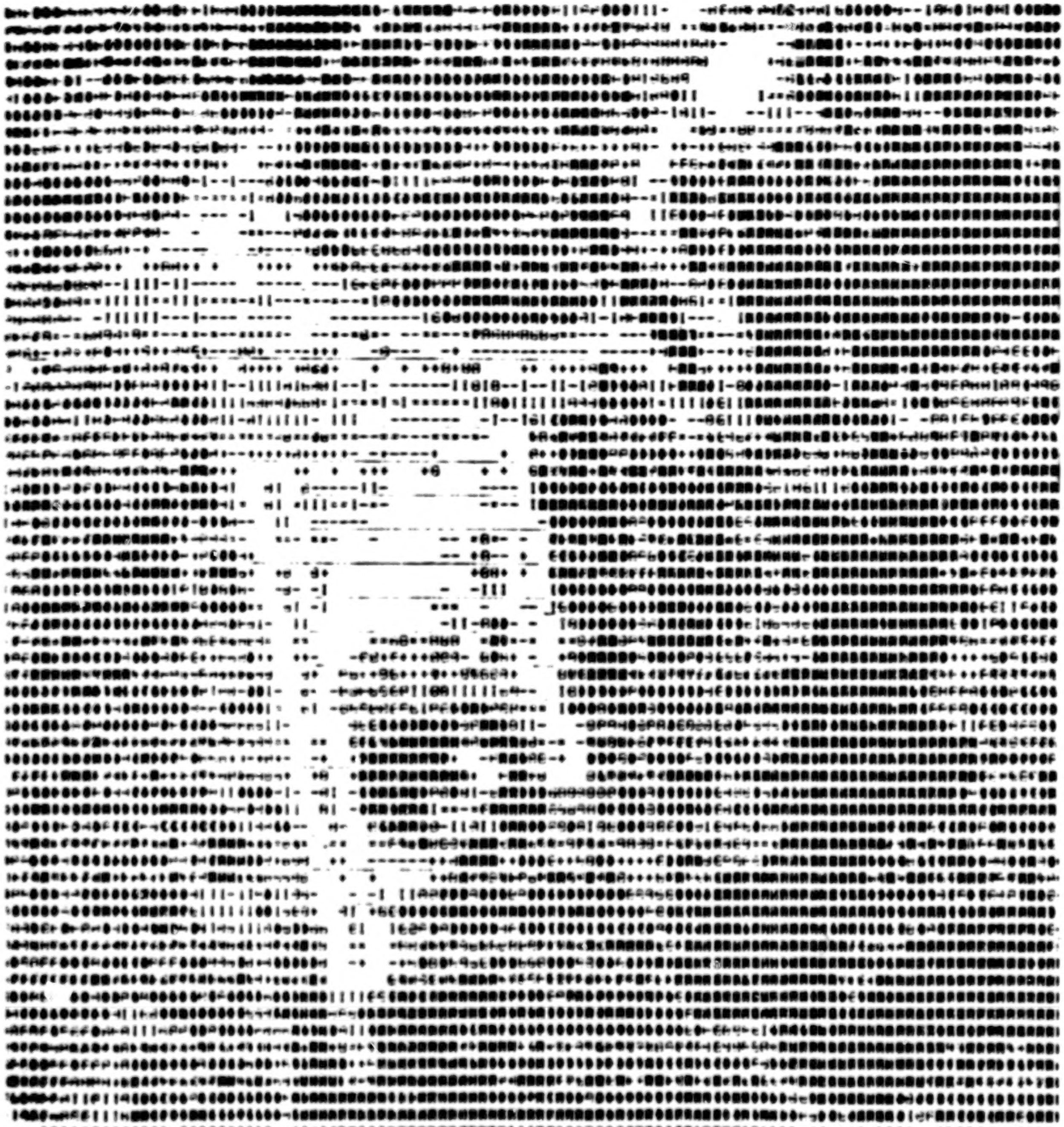


Figure 11. Printer Plot of Landsat Data Coverage of Huntsville Jetport

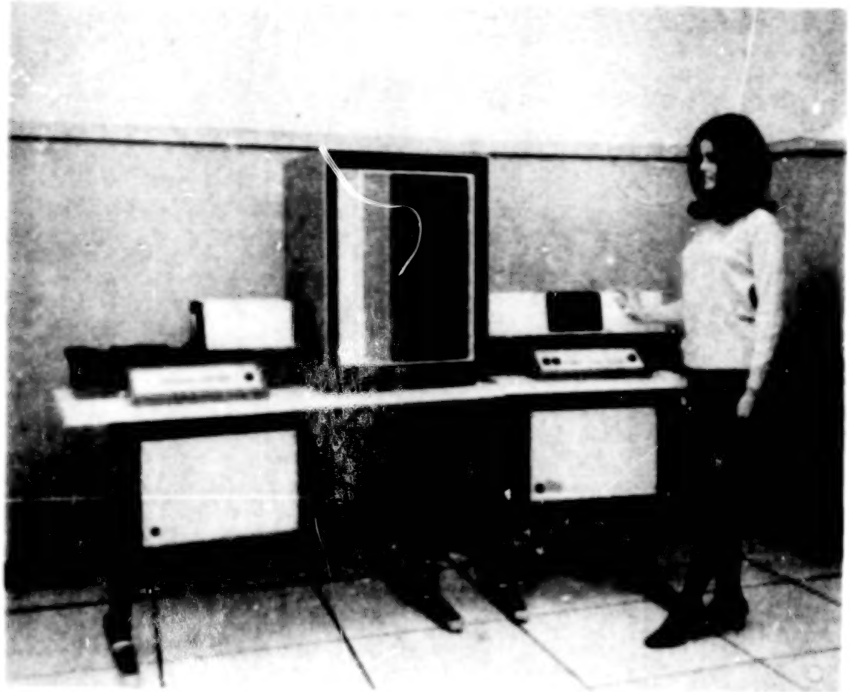


Figure 12. Film Writing and Scanning Equipment with Magnetic Tape Unit

1-5. APPLICATION TO TARCOG AREA

TARCOG (Top of Alabama Regional Council of Governments) is a coalition of the five counties (Limestone, Madison, Jackson, Marshall, DeKalb) located in the extreme northeast corner of Alabama, which was formed to better evaluate and respond to socio-economic problems of the area. One such problem is the land usage of the area, which in cooperation with NASA/Marshall Space Flight Center had been surveyed using low altitude (3000-6000 ft.) aerial photography. In addition, some RB-57 aircraft coverage (60,000 ft.) had been obtained and one frame of three band photography analyzed by digital computer. The present study was initiated in order to evaluate the feasibility of Landsat data analysis of land usage in the TARCOG area, and make comparisons with the low altitude and high altitude aircraft coverage. Figure 13 shows the TARCOG area, and the RB-57 and satellite frame sizes.

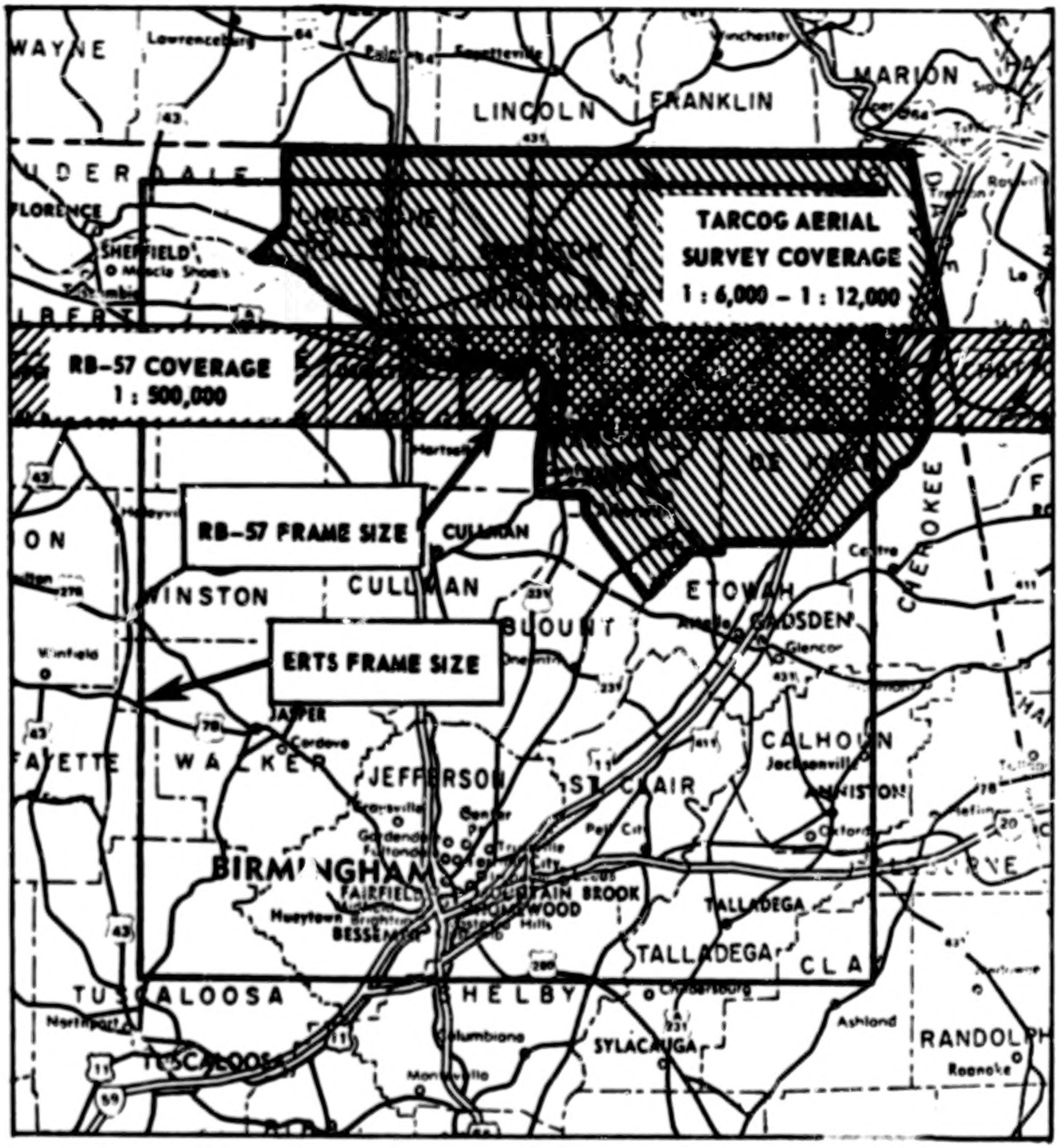


Figure 13. TARCOG Area Showing RB-57 and Satellite Frame Sizes.

II. ANALYSIS PROCEDURES

2-1. PRELIMINARY DATA HANDLING

Multispectral scanner (MSS) data is transmitted to ground-based receiving sites and hence to the NASA Data Processing Facility (NDPF). The NDPF corrects, calibrates and formats the raw MSS data and converts it to a usable binary form on computer compatible tape (CCT). The annotated and corrected 185-km. square ground scene on the CCT is a final product of the MSS. A scene is made up of 2340 parallel scan lines, each containing approximately 3240 data samples.

The NDPF transmits completed ground scenes to data users on four separate CCTs, each containing image data for one 46.25 x 185 km. strip. The images are registered with respect to spectral bands, and are calibrated using a calibration wedge which is introduced into the data during every other scan retrace interval. The CCTs also contain, as part of the annotation record, the geographic coordinates of the image center and the locations of the tick-mark reference system. The tick-marks are located at the intersections of the scene edges with latitudes and longitudes at intervals of one-half degree.

With this degree of geographic coordinate information available, it is possible to construct a computer program which reads the CCT and determines the image coordinates of an area specified by latitude and longitude bounds.

The pixel coordinates are determined using the following five steps:

(i) The pixel coordinates of the format center are found.

(ii) The latitude and longitude of the format center are found.

(iii) The latitudes or longitudes of the tick-marks and their locations relative to the format center are determined.

(iv) Scale factors needed to convert projected latitudes and longitude differences to pixel differences are computed.

(v) Pixel coordinates of the four corners of the area to be extracted are calculated.

The segment of data thus defined may then be transferred to additional magnetic tape to be used in subsequent operations. This output may contain the reflectance measurements from each band in four adjacent storage locations (feature vector format) for each pixel, or may contain all measurements from a spectral band in separate tape files. The feature vector format is most useful when the four spectral measurements are to be considered simultaneously, as in a multispectral classification into land uses.

2-2. COMPUTER CLASSIFICATION

Classification algorithms may be defined as sequences of mathematical operations which determine from a set of measurements the class or type of object which is being measured. When the determination is done by computer, the process is termed automatic feature classification. Generally, measurements of more than one characteristic or feature of the objects in question are made simultaneously. In the present case, the set of measurements is the intensity of sunlight reflected in specified wavelength bands of the visible and near infrared regions of the spectrum. Each set of n measurements is said to define an n-dimensional feature vector, $\{x_1, x_2, x_3, \ldots, x_n\}$, which thus contains all the information obtained by the sensor.

The mathematical criteria which are employed in classifying the feature vectors are called decision functions or discriminant functions. The particular method being used determines the form of these functions. The unknown parameters in these functions are determined in a preliminary process called learning or training. A small part of the data, called the training set, is used by the learning algorithm.

When the classification of the training samples is unknown, the determination of the decision function is said to be unsupervised. The algorithm attempts to find trends in the data and separate the given unknown samples into distinct groups.

Supervised algorithms may be employed when one is supplied with a set of training sample patterns of known classification. These samples are used to develop decision functions, which may then be used to classify unknown samples. The classification will be reasonably accurate if the training samples are truly representative of the classes and an appropriate type of decision function is computed.

Thus, a crucial aspect of the classification problem is the selection of data to be used as training samples. This is generally accomplished by visual inspection of the imagery, coupled with additional sources of information such as topographic maps and personal knowledge of the area. During this process it may be desirable to display the imagery at various levels of magnification, to enhance the imagery by adjusting density levels, and to indicate on the imagery the sites from which data is to be extracted.

In the present study, training data was selected from a region which was a small fraction of the TARCOG area, but which included the city of Huntsville, MSFC, the Huntsville-Madison County Jetport, Monte Sano, and a portion of the Tennessee River. Thus a wide range of land use categories could be defined within a relatively small area. Land usage in this area had been previously studied from aircraft photography, using manual and computer classification

techniques. Thus, it was possible to identify in this section of imagery training sites which represent several land use categories.

The line and sample coordinates of the selected regions may be determined from a CRT display of the imagery, or from a computer line printer display, such as that in Figure 11. The latter has the advantage that individual pixels are readily identifiable.

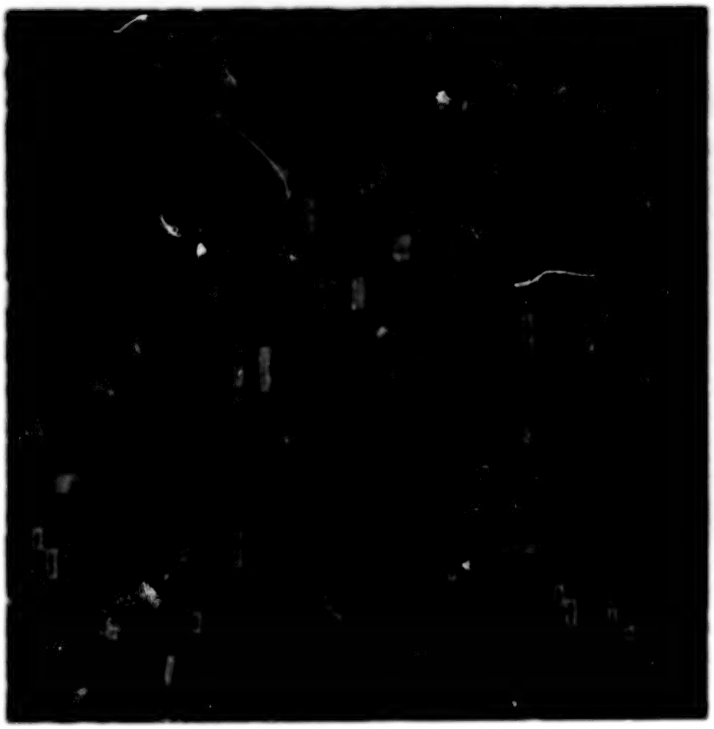
Using the initial training site regions, a classification map may be prepared. The shape and extent of the various land use areas will be more easily seen on the classification map than on the input data. Hence if the training site boundaries are indicated on the classification map, it may be possible to adjust the defining coordinates so that the training data is extracted from areas whose land use corresponds to the desired classes. In addition, misclassifications may indicate that the training data was not sufficiently representative for all areas in the scene. In this case, training data may be extracted from additional regions of the scene.

Locations of the training sites for seven land use classes are shown in Figure 14. The classes and locations are as follows:

Urban	City of Huntsville and Jetport terminal area.
Agriculture	South of Jetport and south of Tennessee River near Highway 231.
Forest	Mountains on east and south of MSFC.
Wetland	Marshes southwest of MSFC.
Pasture	Jones Valley farm and flanking Rideout Road.
Water	Tennessee River
Barren	Quarries northwest of Huntsville.

In separating one class of objects from one or more other classes, it is desirable to de-emphasize the characteristic features that the classes may have in common, and to emphasize where possible the features that are unique to the class of interest. The most obvious first approach is to say that the distinctive character of an object or class of objects is the sum total of its features, some features being more distinctive than others in certain environments. The Linear Classifier concept depends upon this assumption, and aims at developing a single measure of a class's composite features. This measure, the discriminant, is formed by adding the value of each feature (reflectance value or brightness in the case of multiband imagery), after each feature has been weighted according to its usefulness in separating the class of interest from the other classes.

Red Band Data



Classification Map

Figure 14. Locations of Training Data for Seven Land Use Classes

Nonparametric methods are so termed because parameters (such as mean values and covariances) of the distribution functions of the data are not used. The training algorithm determines the values of the weighting factors "w" to be used in a discriminant function of the form

$$G = w_0 + w_1x_1 + w_2x_2 + \ldots + w_nx_n$$

A set of weights is determined for each class of data, the value of a weight reflecting the significance of its associated feature in separating the class from its companion class. Thus for each unknown feature vector, a value of G is obtained for each class.

There are two approaches possible in the application of linear classifiers. In the first, the discriminant functions are designed such that one class may be separated from each of the other classes, pairwise. Then, in determining the class to which a particular feature vector (the reflectance values from one pixel) should be assigned, the value of G is calculated by substituting the values of the feature vector in the discriminant function for each of the classes. The class for which the value of G is largest is the class to which the feature vector is assigned.

In the second approach, the one employed at NASA/MSFC, the discriminant functions are designed such that one class may be separated from all of the other classes considered collectively as one class. Unlike the first approach in which all discriminants are calculated concurrently, here the discriminants are calculated sequentially. Referring to Figure 15, the straight line corresponds to the discriminant function that will separate Class 4 from Classes 1, 2, and 3 taken together. If a given feature vector lies to the right of this line, the discriminant has a positive value and the vector is assigned to Class 4. If it lies to the left of the line, the discriminant has a negative value, and the vector is not assigned to Class 4. Class 4 may then be removed from consideration, and a further test is applied using the discriminant function for Class 3, say. These tests are repeated until the feature vector is assigned to a particular class, at which time testing ceases, and a new unknown feature vector is called in. The sequential nature of testing results in a speed advantage over the parallel procedure employed in the first approach.

The linear classification scheme described here is combined with a feature selection algorithm that determines which of the features of any class are of greatest significance in separating that class from the others. The method of feature selection is based on the concept that the classification is more accurate if

> data values from different classes are widely separated (interclass distance is large), and

- data values within each class are closely grouped (intraclass distance is small).

These effects are illustrated in Figure 16.

The interclass and intraclass distances are computed for each feature by calculating the totals of the separations between all pairs of points in different classes (interclass) and within each class (intraclass). The optimum is obtained when the interclass distance is maximized and the intraclass distance is minimized.

After calculating the criterion for best features (based on separations between training data of the various classes), the feature selection values are combined to yield a value which determines the most easily separable class (Class 4 in Figure 15), for which the discriminant function coefficients (w's) are then computed.

The analysis process in the training phase is illustrated in Figure 17. After the training samples have been selected, they are processed by the feature selection algorithm EFFECT. This determines which class is the most easily separable from all others, and the optimum subset of features (spectral bands) for separating that class. This latter option may be bypassed if not many (three of four for example) spectral bands of data are available, but it is very useful if many bands of multispectral scanner data have been acquired. The discriminant weights for the most easily separate class are then calculated, using the algorithm SNOPAL.

The values of the weights are determined by an iterative procedure. In each iteration, the value of w is changed slightly from its previous value to produce an improved set of weights. Several options are available in the algorithm for terminating the iteration. Once the weights for the most easily separable class have been determined, the training samples for that class are removed from the data set, and EFFECT then determines the next most easily separable class and its optimum feature subset. Then SNOPAL computes the required discriminant function coefficients. This process of identifying an easily separable class and its discriminant, supressing its data and moving on to the next easily separable class, is repeated until a discriminant function has been calculated for all of the classes in the training data set.

The training phase is completed by performing a test classification of all training samples. Ideally, the classifier should assign the training samples to the class from which they were selected by photointerpretation. If the classifier assigns more than a few samples from Class 4 to Class 1, for example, this will suggest an unsatisfactory choice of training samples, and that some of Class 4's training samples were inadvertently selected for Class 1. The choice of training samples must then be revised, and the entire training phase repeated.

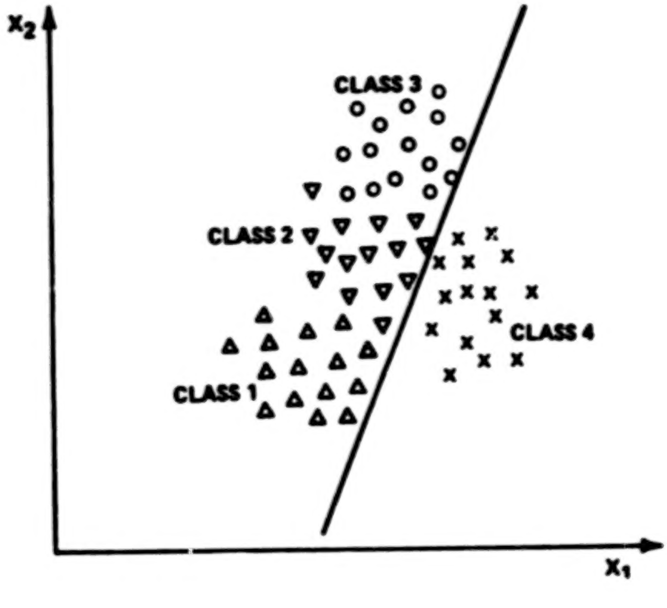


Figure 15. Decision Function for Assigning Samples to Class 4.

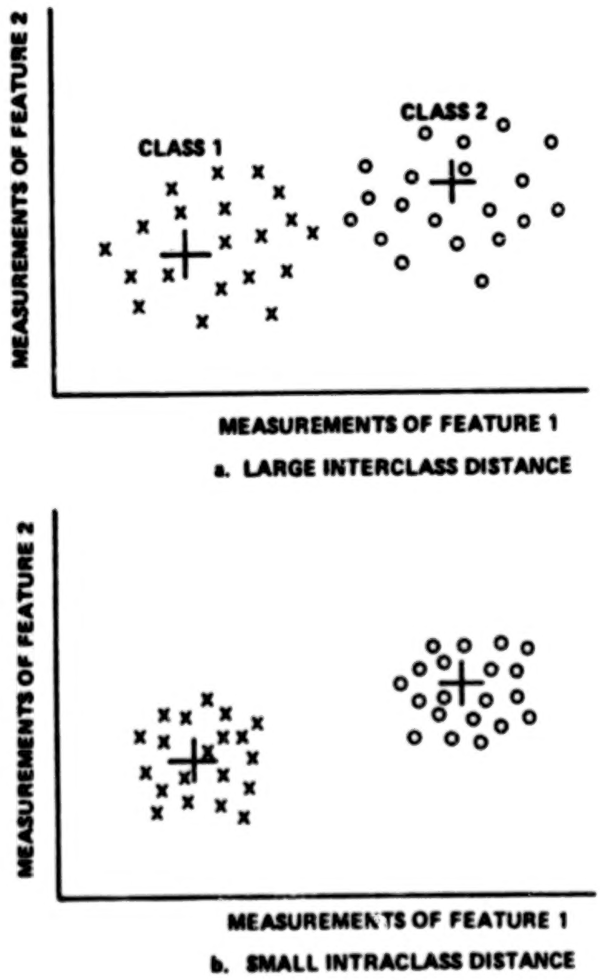


Figure 16. Interclass and Intraclass Distance.

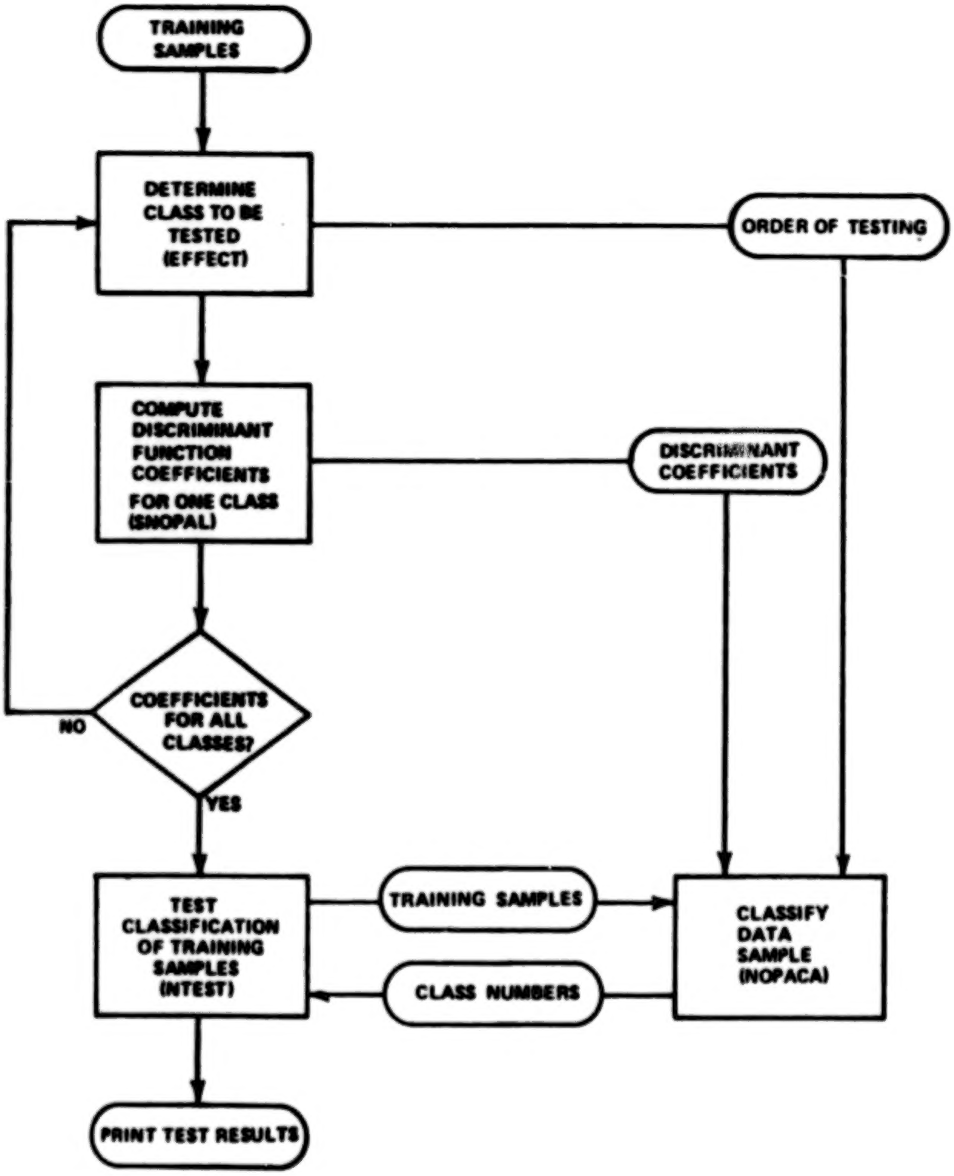


Figure 17. Discriminant Training Phase of Sequential Linear Classifier.

In the classification process for an unknown feature vector, shown in Figure 18, the values "G" of the discriminant functions are computed in the same order as the functions were defined, and the assignment is made to that class for which G first becomes a positive number.

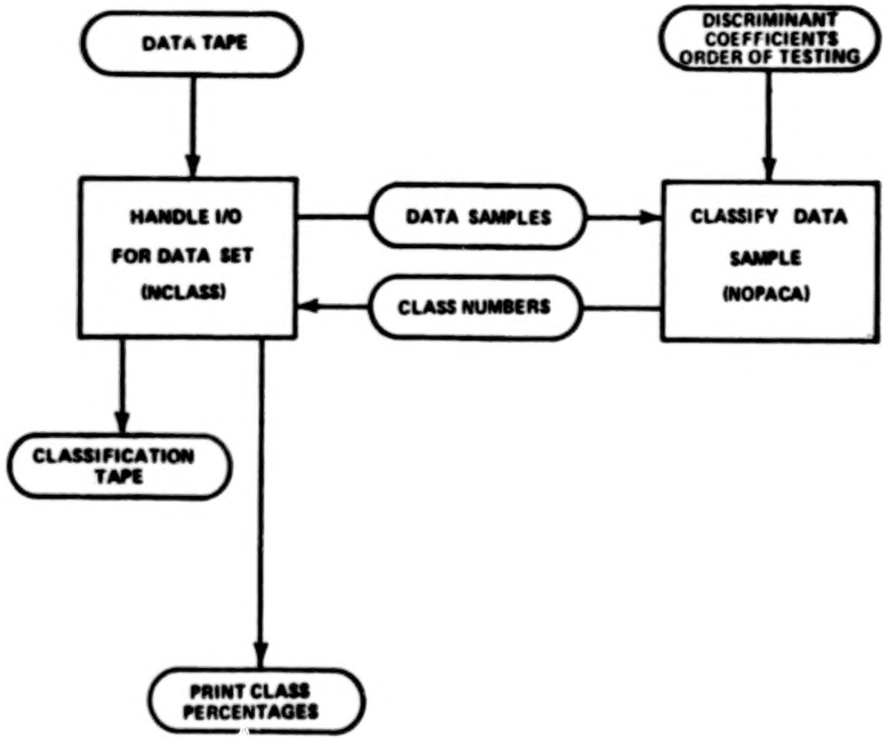


Figure 18. Classification Phase of Linear Classifier.

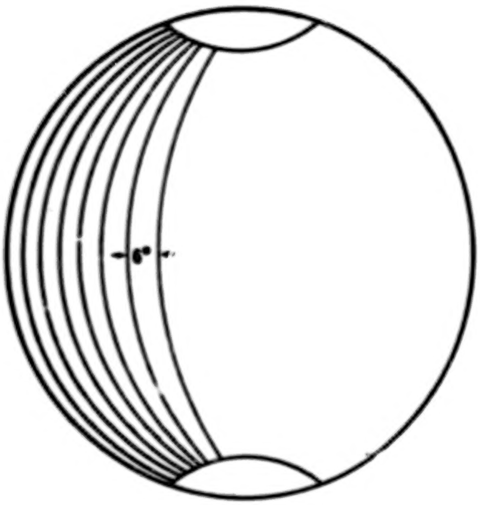
2-3. GEOGRAPHIC REFERENCING

The Landsat-1 system, described in Chapter I, provided the data used in this investigation. Data samples are gathered along scan lines normal to the direction of spacecraft travel, and a ground track of width 185 km. is imaged. However, the resultant image does not correspond to conventioval map-making standards, i.e., equally spaced latitude and longitude lines with longitude vertical and latitude horizontal. In the present situation, there are three principal contributions to the geometric distortion. They are the non North-South heading of the satellite, non-uniform data sampling rates along track and across track, and the rotation of the earth from west to east beneath the satellite. Because for interpretative purposes it is mandatory to associate each Landsat resolution element with a precisely known geographic location on Earth's surface in order to correlate the imagery with aerial photography and existing maps, the distortions in raw imagery must be corrected and the data established in a consistent geographic framework. Then the correlation of sequentially, seasonally or annually observed scenes is greatly simplified, and interpretation errors due to differences of image scale or orientation are minimized. Also standard reference data, for example political boundaries, can be overlaid on the CCT data as a routine processing procedure. Further, the correlation with airborne remote sensors, cameras and line scanners, is simplified by establishing a unified geographic datum. The Universal Transverse Mercator (UTM) projection is used in the present work.

In this system, the surface of the Earth is divided into sixty transverse (i.e. north-south) zones. In international usage these zones are numbered 1 to 60. The center of each zone is called the central meridian. The relation of the UTM zones to the Earth's surface is shown in Figure 19 and the shape of a zone is shown in Figure 20.

Each UTM zone has superimposed on it a rectangular grid of vertical and horizontal lines. The vertical lines lie parallel to the meridian that runs down the center of each zone, and the horizontal lines run parallel to the equator. The basic grid lines are drawn 100,000 meters, or about 62 miles, apart. These grid lines are shown in Figure 21. The squares formed by the intersection of the 100,000 meter lines are usually subdivided by 10,000 meter lines, 1,000 meter lines, or 100 meter lines, depending on the scale and purpose of the map.

The 100,000 meter grid lines are referenced by their "northing" and "easting" values. The northing value is the distance of the line from the equator. Vertical lines are counted from the central meridian which is the 500,000 meter line, those on the left of it having an easting value of less than 500,000 meters and those on the right having a value above that. This is shown in Figure 21.



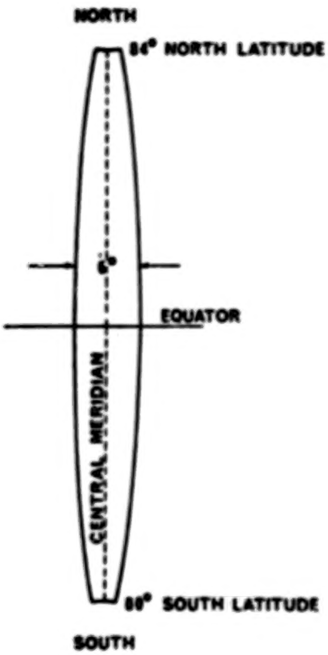


Figure 19. Universal transverse Mercator zones.

Figure 20. Shape of UTM zone.

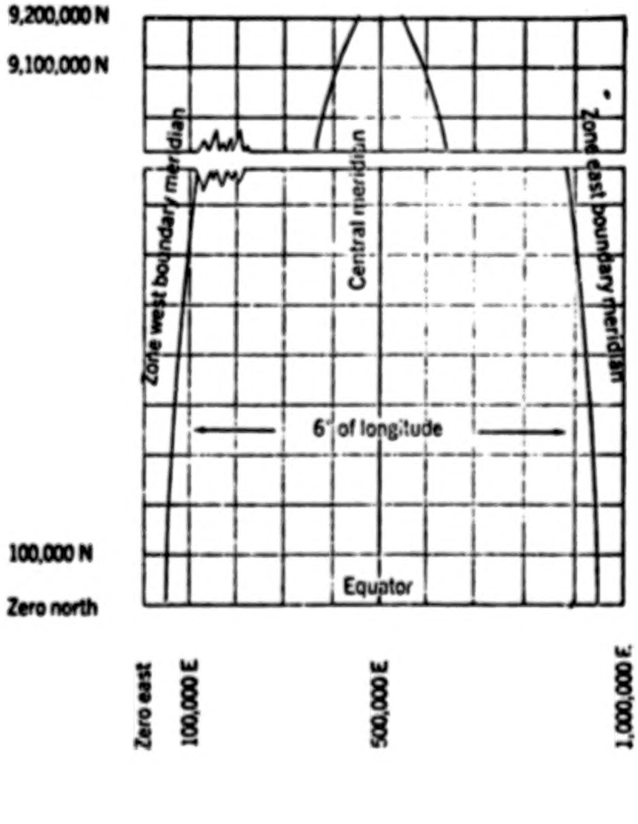


Figure 21. A UTM zone with 100,000-meter grid superimposed.

Two approaches may be taken to the geographic referencing problem. The cartographic approach consists of correcting (or adjusting) the image data to match an Earth coordinate system. This is convenient for the preparation of displays or maps. The other approach systematically warps Earth coordinates to match row and column coordinates in the imagery. In either case, it is necessary to find the equations of transformation between points in the image and locations on the ground.

There are two methods that can be considered for determining the transformations: theoretically, by calculating the effects of all the processes involved in producing the image, and empirically, by comparing the image with a model (e.g., a map) of the terrain. The first requires detailed knowledge of the flight path or orbit, attitude and motions of the sensor-carrying vehicle, characteristics of the sensor, and important error sources. This approach is impractical in all but the simplest cases. This leaves the empirical method. If it is possible to determine the geographic coordinates of every point in the image, an expression for the transformation can be found exactly (assuming the requirements of the sampling theorem are satisfied). In any case, the geographic referencing problem is solved for that image. However in a practical case it is difficult to do more than locate a relatively small number of landmarks in the image, and often virtually impossible to find their <u>exact</u> row and column coordinates. In MSS data, for instance, this problem is accentuated because the instantaneous field of view is large and the resolution is relatively coarse. This suggests using a regression technique to fit a model of the transformation to the landmarks or control points, expressed in both coordinate systems. Very accurate maps are available for the United States and many other parts of the world, from which to obtain the geographic coordinates.

Landmarks or ground control points (GCP) ideally should possess well defined characteristics of shape, should exhibit high contrast against their background in one or more spectral bands, and should not change materially in contrast because of seasonal or climatic influences. Prominent land/water interfaces in the infrared, and man-made constructions such as major highway intersections or airport runways in the green, are well suited as GCP's. To pinpoint the geographic coordinates of a GCP within a Landsat scene, the CCT data is presented for inspection on a digital image display, candidate GCP regions are identified, their data arrays are extracted, magnified by a scaling algorithm to the point that individual resolution elements can be easily seen, and redisplayed. The data address (sample number and scan line number) of a single resolution element (pixel) within the GCP that can be associated with a unique geographical location on a map is then determined. Repeating this procedure for a number of GCP's provides the data required to establish the transformation between the image (pixel) and the Earth coordinate frameworks. If a digital image display is unavailable, this procedure can be followed, although with greater difficulty, using a computer line printer pseudo grey-level plot of the GCP region for pinpointing pixel coordinates.

After identifying the control points in an image and determining their geographic coordinates, it is necessary to find the amount of geometric correction required. In outline the method is as follows:

The coordinates of the control points in both systems (satellite data and UTM coordinates) are found. With the geographic UTM coordinate taken as the independent variable, the equations of transformation give expected values for the coordinates in the image reference frame. In general, it is desirable to use as large a number as available of control points to compensate for errors, if any, between the given image and the standard reference map. Also, it is desirable to perform geometric manipulations with a small number of parameters. This results in a large number of equations to solve for a small number of parameters. More often than not, exact solutions for parameters do not exist in such cases. Therefore the parameters should be determined such that the error between the estimates of the control points' coordinates using the parameters and their exact coordinates is minimized in some sense. The method chosen for solving for the fit parameters was classical Gaussian least squares, modified to work with vector observations. The basic generalization to vector observations consists of replacing the sum (over observations) of the squares of the deviations between model and observations with the sum of the squares of the Euclidean norms of the difference vectors. Then vector components are treated the same as scalar observations in the simpler case.

The procedure followed was as follows:

(i) The geographic coordinates (x, y) of control points are taken as the independent variable, and the picture coordinates (u, v) as the dependent variable (observations),

(ii) The image was displayed on the Dicomed display screen. With the aid of a map, several control points were identified and their approximate u-v coordinate locations were found.

(iii) Previously existing image processing software was used to extract small regions surrounding these approximate locations, magnify them several times by repeating pixels and lines, and format the enlarged regions into a multiple-frame output display.

(iv) The result was viewed on the Dicomed display, and the u-v coordinates of the control points were estimated as accurately as possible. The x-y coordinates were taken from a USGS map.

(v) These coordinates were used in the least squares program to obtain the parameters of the transformation.

2-4. GEOMETRIC CORRECTION

Geometric manipulation of images is needed in handling remotely sensed data in order to match the data obtained by various sensors and/or at several times with respect to a single standard frame of reference. The geometric transformations that need to be implemented may be simple rotations and scaling as in the case of aerial photographs of small regions or combinations of several more complex transformations as in the case of multispectral (linear or conical) scanner output from satellites of large areas on earth wherein the rotation and curvature of earth need be compensated for. The main problem involved in applying the transformations using a digital computer is the bulk of data one has to handle. For instance, in the MSS images there are over 7.5×10^6 bytes of data per frame in each of the four spectral bands. Data is generally supplied on magnetic tapes and the output is required to be on tapes. In contrast with point operations on image densities, geometric manipulation of images generally requires more than one input data record to generate one output record. Also, in many cases it may not be possible to contain all the input records needed to generate one record of output within the main memory of a computer and hence segmentation of input data and reassembly of output records may be required. Further, the sample locations in the geometrically transformed image do not necessarily correspond to integral sample locations in the input image. This requires that some type of interpolation be used for assigning the image intensity values at the output sample locations.

Any geometric distortion of a two dimensional image in a continuous domain may be expressed in the form

$$x' = f(x, y)$$

$$y' = g(x, y)$$

where (x', y') is the location to which the point (x, y) in the image should be moved. Thus a geometric distortion consists in finding (x', y') for every (x, y) in an image and setting the density of the new image at (x', y') to that of the given image at (x, y). Equivalently, when the inverse transformation

$$x = \varphi(x', y')$$

$$y = \psi(x', y')$$

exists one could compute (x, y) for every (x', y') and set the density at (x', y') to that at (x, y) in the given image.

In the case of a digitized image, it is possible that the sample point (x_i', y_j') in the new image does not map into any point (x_k, y_ℓ) on the sampling grid of the original image. Therefore it is necessary to define the image density at (x_i', y_j') in some manner. If the continuous image function is band limited and

the sampling fine enough, the sampling theorem can be used to obtain the exact density at (x_i', y_j'). However, since this is a slow process and there is no guarantee that the sampling frequency is sufficiently large, some simple techniques of interpolation are used instead. Some common approaches are the nearest neighbor rule (causing some geometric uncertainty particularly at boundaries between different types of ground cover), and bilinear or cubic interpolation (leading to radiometric distortion).

Of a variety of models $f(x, y)$, and $g(x, y)$ appropriate for the characterization of Landsat image distortions, it is found that a linear transformation between original and corrected image coordinates compensates the predominant distortion components. Using 23 GCP's, for example, the root mean square compensation error is less than the dimension of the Landsat resolution cell.

2-5. SUPERPOSITION OF BOUNDARIES

In the study of remotely sensed images for land use analysis and planning, it is generally of interest to determine the distribution of land use classes within politically delineated regions such as states, counties or cities. Therefore, it is necessary to first associate the boundary information of the desired type with the remotely sensed images and then extract the region in the interior of a certain political entity as required for further evaluation. In this section we shall describe the steps involved in superposing boundaries on images and separating the image data into individual political entities.

The steps involved depend upon the type of equipment available to digitize the boundary data. The method described below was designed for a system employing a microdensitometer capable of digitizing transparencies. Some of the steps would be obviated if a draftsman's table with a digitizing plotter/tracer attachment were employed.

The steps required when a microdensitometer is used for digitizing are:

(i) Drafting

(ii) Photographic reduction

(iii) Digitization

(iv) Thinning and conversion to "scan line intersection code"

(v) Smoothing to assure continuity

(vi) Finding control point coordinates in pixels and UTM system

(vii) Determination of the required geometric transformation to assure that the image and boundary data are in the same coordinate system

(viii) Application of the geometric transformation

(ix) Thickening of boundary data (if desired) and superposition on image to obtain a combined picture for visual inspection

(x) Identification of separate regions and extraction of data corresponding to each region from the remotely sensed image

A general description of the above steps follows.

(i) Drafting

A standard map of a convenient size is used to obtain the desired boundary lines. The lines are traced in black on a translucent paper. The tracing should be as accurate as possible in order to assure geometric fidelity when matched with the remotely sensed image.

(ii) Photographic Reduction

The tracing is reduced photographically to a transparency of size convenient for digitization on the microdensitometer. (It is preferable to do this rather than try to get a tracing of the size the microdensitometer can handle, since the effects of drafting errors would be more pronounced in a small size tracing.)

(iii) Digitization

The image on the transparency is digitized at a resolution close to or finer than the final anticipated resolution (in km/pixel). This should be done in preparation for the geometric correction step. If the digitization is too coarse, most of the points after correction will have to be generated by interpolation and the resulting image of the boundary will be inaccurate and will show jaggedness, depending on the type of interpolation used.

(iv) Thinning and Conversion to Scan Line Intersection Code

When digitized with a microdensitometer, the data generated are the density values at all pixel locations within a rectangular region on film. Thus, a 25×25 mm^2 region scanned at a resolution of 12.5 μ generates $2000 \times 2000 = 4 \times 10^6$ density values. But, when the image under consideration is a boundary image where most locations are blank and only the positions of a few lines constitute the relevant information, it is more efficient to store and expeditious to handle the boundary points' coordinates. Typically, the boundary lines in the above example may be represented by the coordinates of 10,000 to 20,000 points.

Several methods of boundary encoding are available (see [1], for example). The most convenient method for our purposes is the "scan line intersection code" (SLIC). With this code we represent the digitized boundary image by giving the sample numbers corresponding to the boundary locations in each row. For instance, while storing the boundary information on a tape, each record can be used to represent one scan line, the record consisting of the number of intersections of the scan line with the boundary lines followed by the sample numbers of those intersections arranged in ascending order. The information can be handled in a computer memory by using two arrays, the first array consisting of all the column coordinates corresponding to the boundary intersections and the second array providing a means of finding the bounds on the addresses in the first array of the coordinates corresponding to a given row (scan line). As an example, consider a simple boundary image shown in Figure 22. A digital version of it is shown in Figure 23. Each grid intersection in 23 is a sample location, and those marked with a dot correspond to the boundary pixels. Now, if we were to represent the boundary image by the densities at all sample locations as generated by a microdensitometer (for example), then the array would consist of 169 values (say, 0 for non-boundary points and N for boundary points). The same data can be represented as 13 records shown below, requiring 63 values.

Record No.	Data
1	0
2	0
3	10, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13
4	4, 3, 7, 10, 13
5	4, 2, 7, 10, 13
6	4, 2, 7, 10, 13
7	4, 2, 7, 12, 13
8	3, 2, 6, 13
9	3, 2, 5, 13
10	3, 2, 5, 13
11	3, 2, 5, 13
12	12, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13
13	0

Also, as two arrays in core, the same data can be represented as follows:

Index array: 1, 1, 1, 11, 15, 19, 23, 27, 30, 33, 36, 39, 51, 51

Data array: 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 3, 7, 10, 13, 2, 7, 10, 13, 2, 7, 12, 13, 2, 6, 13, 2, 5, 13, 2, 5, 13, 2, 5, 13, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13

Now, we shall consider the problem of converting the digitized data to the SLIC. To do this, we first need to detect the locations of boundary points. An adequate criterion for this is a threshold on the density values. However, the boundary lines thus detected turn out, in general, to be more than one pixel in thickness. Since in most problems involving boundaries it is desirable to have as thin a boundary as possible, we reduce the thickness of the lines using a thinning algorithm. Referring to Figure 24, the purpose of the thinning algorithm is to generate an approximation to the dashed line given the "thick" lines in digital form. After a thin boundary line is obtained, the coordinate information is converted to the SLIC.

(v) Smoothing

Discontinuities might occur in the thinned boundary data due to drafting and photographic defects or thresholding and thinning. For interior extraction or political entity separation, it is important that the boundary be continuous. Therefore, the thinned data are examined at every point for continuity and patches are generated between locations of discontinuity and the nearest boundary point (if any, within a pre-specified maximum distance).

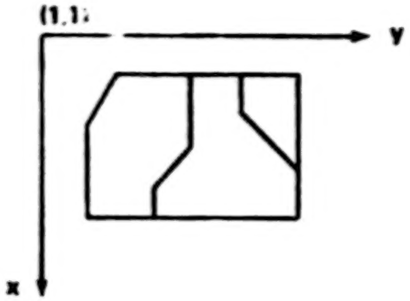


Figure 22. A Simple Boundary Image

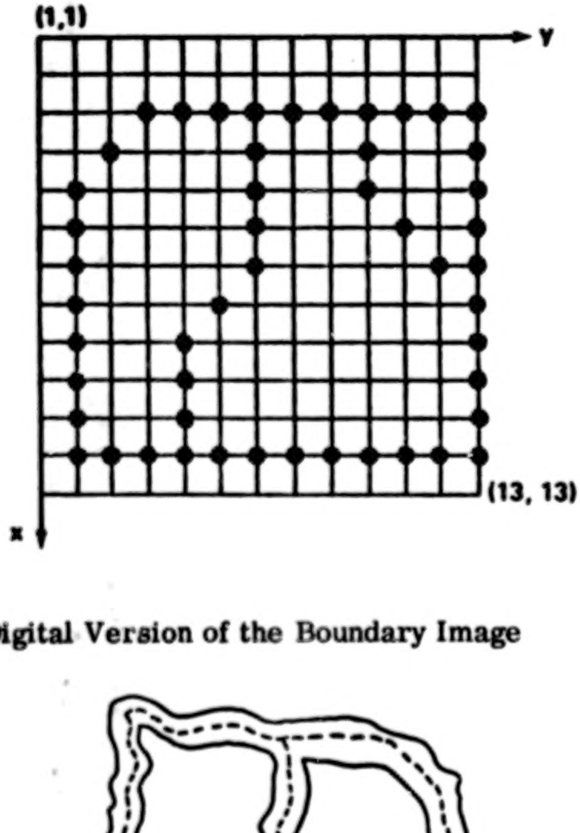


Figure 23. Digital Version of the Boundary Image

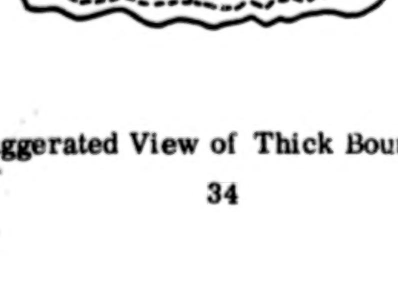


Figure 24. An Exaggerated View of Thick Boundaries

(vi) Finding Control Points

To find the corrections in scale and orientation required for matching the boundary data with a standard coordinate system, it is necessary to establish the coordinates of some known locations called control points. A convenient set of points while handling boundaries are intersections of boundary lines. Figure 25 shows a boundary map of the five-county area (TARCOG, in North Alabama) on which the work was performed. The control points are shown on the map. The ground coordinates of these points can be determined in the UTM system by reference to standard maps. The pixel coordinates of the same points can be determined by obtaining binary line printer plots of small sections of the boundary data including the control points and manually counting the pixel numbers.

(vii) Determination of the Geometric Transformation

The transformation needed to convert from the pixel numbers as obtained from the microdensitometer to the standard coordinate system (at a specified sampling interval) can be found from a knowledge of the control point coordinates. A parametric model is assumed depending on the types of correction required. A linear transformation with six parameters is sufficient to account for translation, rotation, and scale change. The parameters are then determined by minimizing the mean squared error between the observed UTM coordinates and those obtained by converting the pixel coordinates using the assumed transformation.

(viii) Application of the Geometric Transformation

The boundary data are converted to the UTM coordinate system by using the transformation determined as mentioned above. A resampling problem enters into the picture at this point. The boundary points which have integer coordinates in the original system do not necessarily transform into integer sample numbers in the UTM system. Therefore, the transformed coordinates are approximated by rounding them off to the nearest integers. Also, when there is a scale change, the number of boundary points in the output image is not necessarily the same as that in the input. The points that were contiguous in the input image might transform into non-contiguous points. Thus, to preserve continuity, it is sometimes necessary to interpolate and generate extra boundary points. A simple approach to this is to transform all the input boundary points via the given transformation, join the output points corresponding to contiguous input points by straight lines and obtain integer coordinate values by rounding off.

(ix) Thickening and Superposition

Whereas, to extract a region within a given boundary, it is necessary to have as thin a line as possible, for visual presentation of boundaries on remotely sensed images, it is desirable to thicken the boundary lines. The data in the SLIC format can be conveniently used to generate thickened boundary data in the same format by producing new boundary points at locations surrounding each old

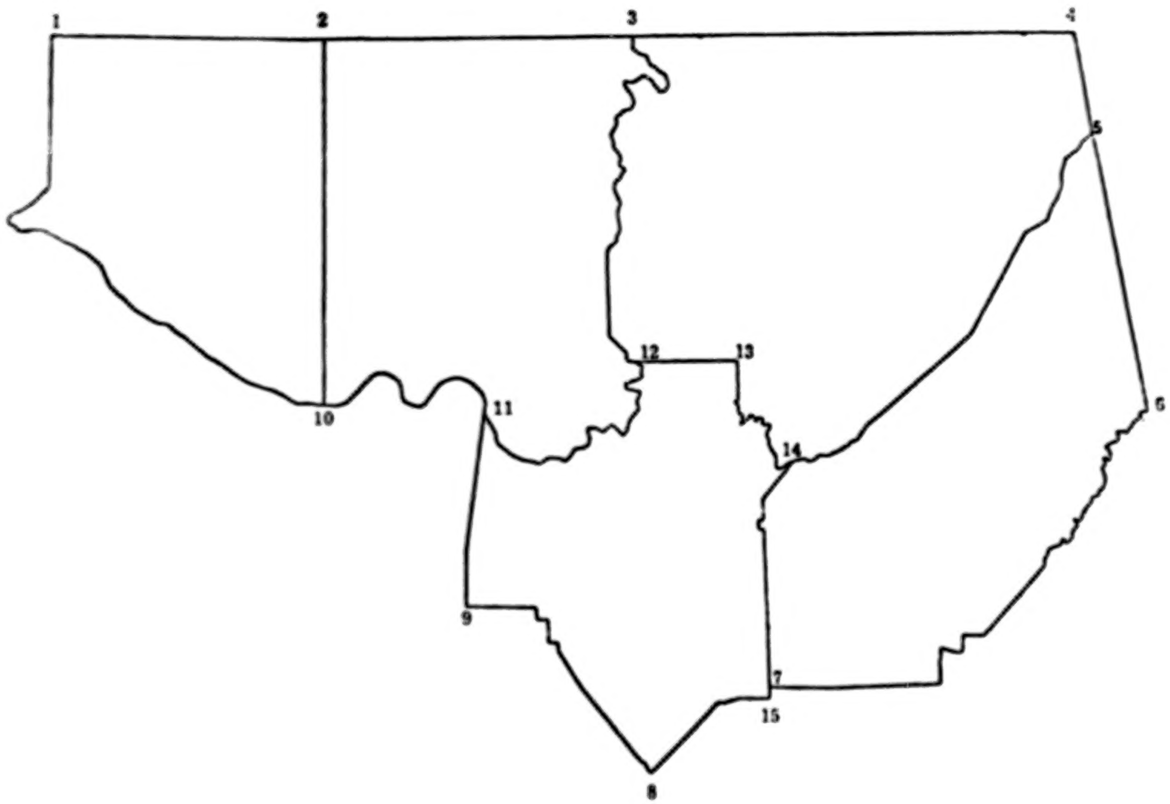


Figure 25. Boundary Map of Five TARCOG counties showing Control Points.

boundary point up to a given thickness. The boundary lines are superposed on the given image (which is of the same scale and orientation) by assigning a unique density to all points in the image corresponding to the boundary point coordinates.

(x) Identifying and Extracting Individual Regions

Each political entity can be extracted separately using the boundary data for the entire region in SLIC format after geometric correction. The first step in doing this is to identify connected regions separated by the boundary lines and generate a unique label for each of the regions. For example, the digital boundary image shown in Figure 23 leads to a "region identification map" (RIM) shown below.

Record No.													
1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	1	1	1	1	1
3	1	1	1	0	0	0	0	0	0	0	0	0	0
4	1	1	0	2	2	2	0	3	3	0	4	4	0
5	1	0	2	2	2	2	0	3	3	0	4	4	0
6	1	0	2	2	2	2	0	3	3	3	0	4	0
7	1	0	2	2	2	2	0	3	3	3	3	0	0
8	1	0	2	2	2	0	3	3	3	3	3	3	0
9	1	0	2	2	0	3	3	3	3	3	3	3	0
10	1	0	2	2	0	3	3	3	3	3	3	3	0
11	1	0	2	2	0	3	3	3	3	3	3	3	0
12	1	0	0	0	0	0	0	0	0	0	0	0	0
13	1	1	1	1	1	1	1	1	1	1	1	1	1

Here, 0 is used for boundary points, 1 for the exterior and 2, 3, and 4 identify the interior of the three separate regions. Such a map is easy to generate from the boundary data in SLIC format using tests for connectivity.

A RIM can be used to extract data corresponding to any given region from a remotely sensed image. For example, all points in region 2 can be highlighted by reading the RIM and setting all densities to 0 except where the RIM values are 2.

III. MATHEMATICAL TECHNIQUES

3-1. PRELIMINARY DATA HANDLING

This section describes the extraction of a geographic location from a computer compatible tape (CCT), given the latitudes and longitudes bounding the area. Since the scanning direction is not parallel to latitudes or longitudes, the smallest rectangular region containing the desired part of the image is determined using the identification and annotation data read from the CCT.

The first step is very simple. The number of records in the CCT is a constant equal to 2340. The number n of pixels per record (per band) is given by "MSS adjusted line length" contained in the 39th and 40th characters [2]. Thus the pixel coordinates of the format center are given by (1170.5, (n+1)/2).

In order to find the bounds on the region to be extracted in terms of pixel coordinates, it is sufficient to determine the pixel coordinates of the four corners of the rectangle bounded by the given latitudes and longitudes. Therefore, we shall describe the method for determining the pixel coordinates of a given point where its latitude and longitude are given.

Let E-N and R-P represent the geographic and pixel coordinates of a given point. Let (e_o, n_o) and $r_o, p_o)$ be the corresponding coordinates of a reference point, say the format center. The satellite heading is given by the angle θ between the N-axis and the R-axis (see Figure 26).

From the equations for rotation of a Cartesian coordinate system about the origin, it follows that

$$r - r_o = (n - n_o)\cos\theta + (e - e_o)\sin\theta$$

$$p - p_o = -(n - n_o)\sin\theta + (e - e_o)\cos\theta$$

if the units of measurement are the same for the two coordinate systems. (An approximation is made here in that the longitudes are assumed parallel to one another - a reasonable assumption if the region to be extracted is sufficiently small and sufficiently far from the poles. Hence, the use of plane geometry instead of spherical trigonometry.)

The scale factors required to convert the distances $(r-r_o)$ and $(p-p_o)$ are computed as follows. The tick-marks indicate the intersections of known longitudes or latitudes with edges parallel to the pixel coordinate axes. Also the number of pixels (records) between two tick-marks along a top or bottom (left or right) edge can be determined. Therefore, the number of pixels (records) per degree of change in longitude (latitude) in the horizontal (vertical) direction

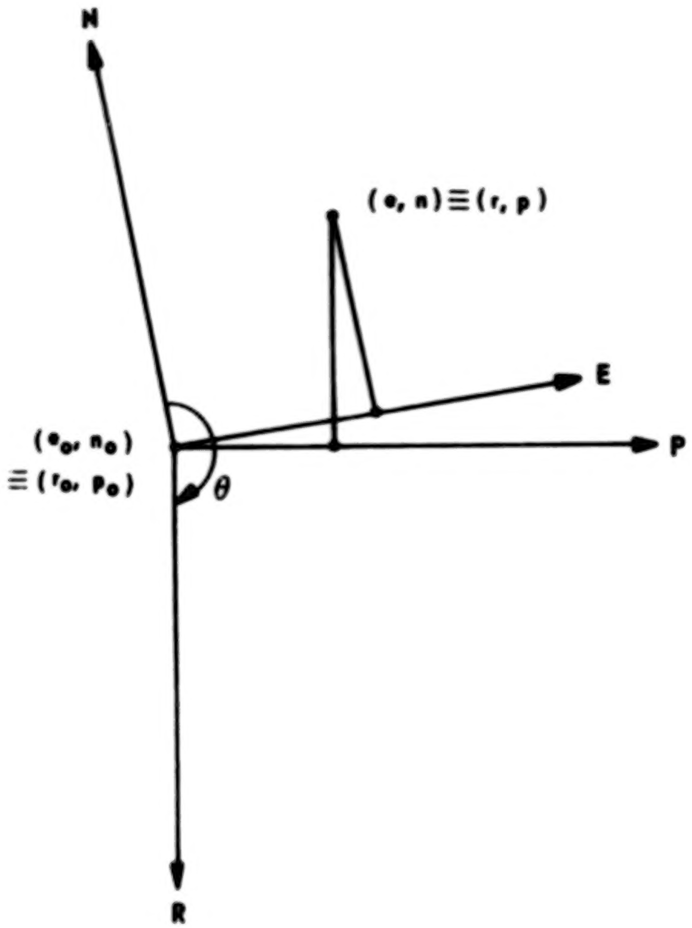


Figure 26. Geographic and Pixel Coordinate Systems

can be determined from the tick-mark data. These then are the scale factors to be used to convert $(r-r_0)$ and $(p-p_0)$ into increments in record and pixel numbers.

The procedure is to first read the ID record, find r_0 and p_0, then read the annotation record to find n_0 and e_0, and the tick-mark information and convert them to floating point numbers, compute the scale factors and find the heading θ and use the formulas above and the scale factors to find the record and pixel numbers corresponding to each of the four corners of the rectangular region to be extracted. Finally, these coordinates are converted into integers, the smaller of the bounds being truncated and the larger being rounded to the next higher integers.

3-2. COMPUTER CLASSIFICATION

The linear discriminant functions presently being used to separate classes of data divide a set of data into two regions, arising from the positive and negative results obtained when individual data samples are substituted into the discriminant function. Thus, as a starting position, it is necessary to define that class of data which can be separated from the remainder of the data by the appropriate decision function. Using the set of training data, the separation between classes is determined according to the following conditions:

1. the two clusters of class data values are widely separated (interclass distance (S_1) is large), and
2. data values within each class exhibit low dispersion, i.e., are closely grouped (intraclass distance (S_2) is small).

In the ideal case, the intraclass distance is negligible compared to the interclass distance, i.e., $S_2/S_1 \rightarrow 0$. If the data samples of different classes fall in the same region, $S_2/S_1 \rightarrow 1$. In the extremely poor case, where data values of class 2 all lie within the extreme data values of class 1, the intraclass distance (S_2) for the more widely dispersed class 1 is greater than the interclass distance (S_1) to the data values of class 2, and $S_2/S_1 > 1$. A normalized figure of merit for assessing the discriminatory effectiveness of a given feature is defined as

$$F = \mathrm{Exp}\ [-S_2/S_1]$$

an index that is suitably bounded between 0 and 1.

The generalized distances S_1 and S_2 are based on the average distances between all pairs of data samples in the class or classes involved. As a first step in the computation, an array D is defined whose elements are the interclass and intraclass distances along each feature. Assuming that an N-dimensional feature vector $X = \{X_i\}$, $i = 1 \ldots N$, defines the sample measurements over M classes, the dimensions of the array are M x M x N. For classes p and q, containing n_p and n_q samples, respectively, and considering feature f, the array element corresponding to the distance between classes p and q along feature axis f is

$$D_{p,q,f} = \sum_{i=1}^{n_p} \sum_{j=1}^{n_p} \left| X_{p,f,i} - X_{q,f,j} \right| .$$

This distance element comprises a total of $n_p n_q$ terms. The diagonal elements of the distance array are the intraclass distances and are defined as

$$D_{p,p,f} = \sum_{i=1}^{n_p} \sum_{j=1}^{i} | X_{p,f,i} - X_{p,f,j} | .$$

Each such element comprises $n_p(n_p-1)/2$ nonzero terms. The various elements of the array D are combined to form the total inter- and intraclass distances among the classes under consideration.

In the present scheme as already outlined, a linear discriminant is to be identified that will separate any one class from all of the remaining classes taken together, and the process is repeated with the number of classes under consideration being reduced by one each time. Thus, the problem of separating M classes is reduced to (M-1) two-class problems in which each discriminant hyperplane successively partitions the sample space. If class p is under consideration, the second class (q) consists of all the remaining classes (q_1, q_2, q_3...) considered together. These original data classes are now, in effect, subclasses of the class q.

The total interclass distance for feature f is then the sum of the distances between class p and each of the subclasses and is defined by

$$\Sigma_{1,p,f} = D_{p,q_1,f} + D_{p,q_2,f} + D_{p,q_3,f} + \cdots$$

The total number of individual distance terms is $n_p(n_{q_1} + n_{q_2} + n_{q_3} + \ldots) = n_p n_q$ where $n_q = n_{q1} + n_{q2} + n_{q3} + \ldots$ and hence the average interclass distance from class p to all other classes along feature axis f is given by

$$S_{1,p,f} = \Sigma_{1,p,f}/n_p n_q$$

The interclass distance for class p itself is simply the array element $D_{p,p,f}$ which is the sum of $n_p(n_p-1)/2$ terms.

For class q, the intraclass distance is the sum of all the distances involving the subclasses q_1, q_2, q_3 ..., namely

$$\begin{aligned} D_{q,q,f} = D_{q1,q1,f} &+ D_{q1,q2,f} + D_{q1,q3,f} + \cdots \\ &+ D_{q2,q2,f} + D_{q2,q3,f} + \cdots \\ &+ D_{q3,q3,f} + \cdots \\ &+ \cdots \end{aligned}$$

This expression is the sum of $n_q(n_q-1)/2$ terms.

The total average intraclass distance for the two classes, therefore, is

$$S_{2,p,f} = D_{p,p,f}/n_p(n_p - 1) + D_{q,q,f}/n_q(n_q - 1)$$

and the figure of merit for determining the effectiveness of feature f in separating class p from all other classes is

$$F_{p,f} = \mathrm{Exp}\,[-S_{2,p,f}/S_{1,p,f}].$$

An M x N array is computed using this expression giving a figure-of-merit matrix that exhibits the effectiveness of all features in separating each of the classes present.

In parallel with this merit figure evaluation, a further computation determines the figures of merit between class p and each of the subclasses q_1, q_2, q_3 ... The purpose of this calculation is to determine whether any of the subclasses are poorly separable from class p, even though the figure of merit of separating class p from class q may have a high value. This can occur when several of the subclasses are very well separated from class p and, hence, heavily weight the value of $F_{p,f}$ while at the same time one or more of the subclasses q_i are poorly separated from p and, hence, the classification ambiguity between class p and these particular subclasses q_i would be considerable. The smallest of these individual figures of merit is multiplied by the overall figure of merit defined above. Thus, the final figure of merit contains two factors:

1. The separability of class p from the remaining classes considered together as the second class,
2. the separability of class p from the nearest neighboring class.

In order to determine the order of separability of the training classes, the figures of merit for individual features of a class are combined to form a single figure of merit for that class. By ordering these values according to magnitude, the most easily separable class is identified.

The determination of the linear classifier discriminant functions is discussed in the following section.

The problem of designing a pattern classifier may in general be expressed as one of determining the discriminant functions $G_i(x)$, $i = 1, \ldots M$, such that for any pattern sample vector X_j, the inequality

$$G_i(X_j) \geq 0$$

implies that X_j belongs to pattern class C_i.

The structure of the classifier is dependent upon the functional forms of the discriminants and also upon the availability of a sufficient quantity of a priori data that adequately characterize representative samples of the patterns to be classified. As long as the assumption may be justified that the pattern classes can be separated by a linear hyperplane, a linear discriminant function leads to the simplest structure of classifier. In this case, a variety of techniques exists for determining the actual structural properties of the classifier. [3]

In multiclass problems, a linear classifier may be applied in either a parallel or a sequential mode. In the parallel mode, the discriminant functions for a given sample vector are computed simultaneously, the largest resulting value identified, and sample assignment is made to the class corresponding to the largest discriminant. The resulting classifier is cumbersome, since the discriminant for any one class must be capable of separating that class from each other class taken individually, and requires $M(M-1)/2$ linear segments when M classes are present. In the sequential mode, the classifier structure is simpler since sample classification into M categories is performed by a sequence of (M-1) dichotomies, and each discriminant is required only to separate its corresponding class from all other classes taken together.

It is well known that a dichotomous linear classifier or Threshold Logic Unit (TLU) defined by the discriminant function

$$G(x) = w_o + \sum_{i=1}^{N} w_i x_i$$

exhibits the following properties: [4]

1. The classifier separates patterns by a hyperplane decision surface in measurement space.

2. The hyperplane has an orientation given by the weight values $w_1, w_2, \ldots w_n$.

3. The hyperplane has a position proportional to w_o.

4. The distance from the hyperplane to an arbitrary pattern vector X_j is proportional to the value $G(X_j)$.

Given two distinct classes of patterns, therefore, classifier structural design reduces to the problem of determining (a) the orientation and (b) the position of the separating hyperplane. In general, these quantities must be derived iteratively from information contained in the distances of misclassified samples from a trial hyperplane.

In the system reported here, the discriminant functions are determined by employing a gradient procedure, the Ho-Kashyap algorithm, [5, 6] that iteratively minimizes the least-squared classification error over the representative sample classes or training classes.

In the case in which two pattern classes p and q are present, containing respectively n_p and n_q N-dimensional pattern vectors X, the discriminant function is

$$G(X_{pj}) = w_o + w^T X_{pj} = d_{pj}, \quad j = 1 \rightarrow n_p$$

and

$$G(X_{qj}) = w_o + w^T X_{qj} = d_{qj}, \quad j = 1 \rightarrow n_q$$

where

w^T = transpose of the hyperplane weight vector $(w_1, w_2 \ldots w_N)$.

The values d_{pj} and d_{qj} are measures of the perpendicular distances of the respective sample patterns from the separating hyperplane. The above $n_p + n_q$ equations may be expressed as

$$
\begin{bmatrix}
1 & X^T_{p1} \\
1 & X^T_{p2} \\
\vdots & \vdots \\
1 & X^T_{pn_p} \\
\hline
-1 & -X^T_{q1} \\
-1 & -X^T_{q2} \\
\vdots & \vdots \\
-1 & -X^T_{qn_q}
\end{bmatrix}
\begin{bmatrix}
w_o \\ w_1 \\ \vdots \\ w_N
\end{bmatrix}
=
\begin{bmatrix}
d_{p1} \\ d_{p2} \\ \vdots \\ d_{pn_p} \\ \hline -d_{q1} \\ -d_{q2} \\ \vdots \\ -d_{qn_q}
\end{bmatrix}
$$

or, more compactly,

$$A\,\alpha = d$$

where the matrix A of dimensions $(n_p + n_q) \times (N + 1)$ defines the entire set of patterns, the $(N + 1)$ vector α defines the separating hyperplane, and the $(n_p + n_q)$ vector d defines the pattern-hyperplane separations, to within a normalization factor. The feature vectors of class q are negated to ensure $A\,\alpha > o$.

The components of d are positive or zero in the ideal case of totally separable patterns although in practice this condition is unattainable because of misclassification due to imperfectly separate pattern clusters. The optimum hyperplane, however, will minimize the number of misclassifications, i.e., will minimize the number of elements of d having incorrect sign, and will therefore minimize a classification error vector $e = (\beta - d)$, where β is a $(n_p + n_q)$ vector of positive constants. The minimization criterion, of course, is arbitrary, but a quadratic criterion is advantageous since a steepest-descent minimization procedure results in a linear recursion relationship. Therefore, let

$$J = 1/2\,||\,\beta - d\,||^2 = 1/2\,||\,A\alpha - \beta\,||^2$$

The condition for minimum J is given by

$$\partial J/\partial\alpha = A^T\,[A\alpha - \beta] = 0, \;\beta > 0$$

and for a given β, the corresponding hyperplane is determined by

$$\alpha = [A^T A]^{-1} \ A^T \beta$$

Since β is initially an unknown positive vector, it must be determined iteratively from the relation

$$\beta(k+1) = \beta(k) + \delta\beta; \quad \beta(0) \text{ arbitrary}, \ k = \text{iteration index.}$$

Logically, to minimize J, the iteration increments $\delta\beta$ should be proportional to the gradient $\partial J/\partial\beta$. Since

$$\partial J/\partial\beta \,|_{\beta(k)} \quad = \quad \beta(k) - A\,\alpha\,(k)$$

several possibilities arise due to the constraint $\beta > 0$.

1. $\partial J/\partial\beta \,|_{\beta(k)} = 0$, then $\beta(k) - A\,\alpha\,(k) = 0$, the ideal solution

2. $\partial J/\partial\beta \,|_{\beta(k)} > 0$, i.e., $\beta(k) - A\,\alpha\,(k) > 0$, hence an increment $\delta\beta$ will tend to increase the classification error vector, and preferably $\delta\beta = 0$.

3. $\partial J/\partial\beta \,|_{\beta(k)} < 0$, i.e., $\beta(k) - A\,\alpha\,(k) < 0$, hence a positive increment $\delta\beta$ proportional to the gradient may be made.

The rationale for incrementing β therefore is

$$\delta\beta = \rho \left\{ A\,\alpha\,(k) - \beta(k) + |\,A\,\alpha\,(k) - \beta(k)\,| \right\} \begin{array}{l} = 0, \ \partial J/J\beta > 0 \\ = 2\rho, \partial J/\partial\beta < 0 \end{array}$$

where ρ is a positive constant vector.

The Ho-Kashyap training algorithm thus may be summarized as follows

1. $\alpha\,(0) = [A^T A]^{-1}\, A^T\beta\,(0)$; $\beta\,(0) > 0$, otherwise abribrary

2. $\beta(k+1) = \beta(k) + \rho \left\{ A\alpha(k) - \beta(k) + | A\alpha(k) - \beta(k) | \right\}$

3. $\alpha(k+1) = \left[A^T A\right]^{-1} A^T \beta(k+1)$

$$= \alpha(k) + \rho \left[A^T A\right]^{-1} A^T \left\{ A\alpha(k) = \beta(k) + |A\alpha(k) - \beta(k)| \right\}$$

The convergence properties and other details of the algorithm have been discussed elsewhere. [6]

3-3. GEOGRAPHIC REFERENCING

Aside from simple rotation and scaling, there are three categories of geometric distortions that may be present in remotely sensed image data. First, there are effects due to geometry. Primarily, these are the result of projection of features from the curved surface of the earth into the image plane. This may also include the map projection involved; the map coordinate system of primary interest in this document is the Universal Transverse Mercator (UTM) projection. The point of view incorporated in the mathematics to be developed may be illustrated by assuming that a set of geographic grid lines are painted on the ground and transformed into image coordinates by the sensor (and by the equations to be developed). Other distortions are due to dynamics - the motion of the satellite, and rotation of the earth. Then, there may be distortions introduced by the instrumentation. For example, in the case of a scanning imager the relationship between position in the projective image plane along a scan line and the data stream itself may not be linear. (In fact, the ground trace of a scan line may not be a straight line.) Another possible instrumentation effect is a direction-dependent scale factor.

It will be seen that the distortions produced by some of these causes are considerable, while others are (more or less) negligible. Fortunately, the big distortions will also turn out to be the easiest to solve for. It will be seen that a simple mathematical model of the coordinate transformation provides accuracy high enough for many uses.

It will be assumed henceforth that geographic coordinates of points on the earth's surface are in the UTM system. Many projections are used in an effort to display the curved surface of the earth on a flat map, all necessarily involving some distortion. Mercator projections have several useful properties. For one, they are conformal. So, taking any small area, the shape of the regions is the same as on the globe. (The shapes of large areas are distorted because the scale is position-dependent.) Also, standard Mercator projections are the only ones in which lines of constant compass heading (rhumb lines or loxodromes) appear as straight lines. This makes them useful in navigation. A standard Mercator projection is related to a projection from the earth's surface onto a cylinder tangent at the equator. Parallels are horizontal and meridians are vertical. Meridians are equally spaced, while the spacing between parallels veries as the secant of the latitude.

The transverse Mercator projection turns the projection system (or the earth) 90^o. It is related to a horizontal cylinder tangent along a meridian. A standard meridian great circle replaces the equator, and the zone on either side of that meridian is fairly well represented. The UTM system is a collection of transverse Mercator projections. In the UTM system, the earth is divided into 60 zones bounded by meridians whose longitudes are multiples of 6^o west or

east of Greenwich. The zones are numbered sequentially, beginning with 1 for the zone from 180° W. to 174° W., and proceeding eastward. The origin of coordinates for each zone is at the intersection of the central meridian of the zone and the equator. Distances in UTM coordinates ("easting" and "northing") are measured in meters. The central meridian is given a "false easting" of 500,000 meters so all easting coordinates are positive. There is no false northing in the Northern Hemisphere; in the Southern Hemisphere a false northing of 10,000,000 meters is assumed. The latitude limits are 80° N. and S. (A different projection must be used in polar regions.) UTM coordinates of a point on the earth's surface consist of the zone number and the easting and northing coordinates.

The equations sought to account for the geometrical effect of projection would relate picture coordinates to UTM coordinates. However, the mathematics involved is quite intractable, and it has not been found possible to obtain such equations except in a form whose complexity conceals their content. It is somewhat easier to write a sequence of equations describing the situation. The following equations are taken from Reference 7. (Although they refer to satellite observations, only a change in terminology is needed to apply them to aircraft.) They relate latitude ϕ and longitude λ to x and y, Cartesian image coordinates with origin (corresponding to the satellite subpoint) at the center of the image. The subscript SP will be used to refer to the subpoint (picture center), and P will be used to designate the coordinates of an arbitrary point. Figure 27 illustrates the situation being described. A sperical earth is assumed. Also, the line of sight of the sensor is assumed to be straight downward.

The plane of the Landsat orbit is inclined at an angle of 8.906° (0.1554 radian) from a polar orbit. With reference to Figure 28, the equatorical inclination i is 81.094°. Because of this inclination, the satellite crosses meridians of longitude with increasing frequency and at increasing angles at the higher latitudes. The heading of the satellite relative to the local longitude line (azimuth) is obtained by applying the law of sines to the shaded spherical triangle in Figure 28. Noting that the two sides which are also longitude lines have arc lengths related to angles of 90° and $90° - \phi_{sp}$, we have

$$\frac{\sin(180 - H)}{\sin 90} = \frac{\sin \epsilon}{\sin(90 - \phi_{sp})}$$

or

$$\sin H = \frac{\sin \epsilon}{\cos \phi_{sp}} = \frac{0.1548}{\cos \phi_{sp}}$$

The following sequence of equations relate the subpoint latitude and longitude $(\phi, \lambda)_{sp}$ to Cartesian coordinates in the image plane, (x, y).

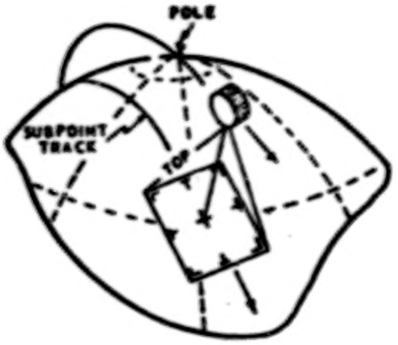


Figure 27. Orientation of picture along Subpoint Track or Heading Line. [7]

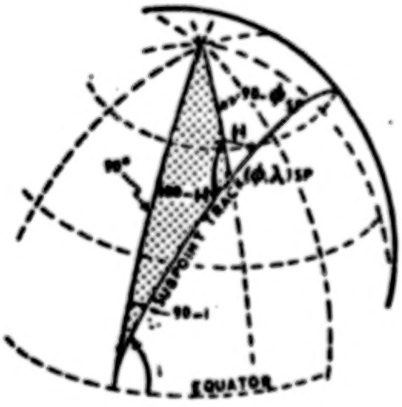


Figure 28. Azimuth of the Heading Line. [7]

Applying the law of cosines to the spherical triangle in Figure 29, we obtain

$$\cos\delta = \cos(90-\phi_{sp})\cos(90-\phi_p) + \sin(90-\phi_{sp})\sin(90-\phi_p)\cos\Delta\lambda$$

$$= \sin\phi_{sp}\sin\phi_p + \cos\phi_{sp}\cos\phi_{sp}\cos(\lambda_p - \lambda_{sp})$$

and by applying the law of sines, $\sin\alpha = \dfrac{\cos\phi_{sp}\sin(\lambda_p - \lambda_{sp})}{\sin\delta}$.

The transformation from (δ,α) to the nadir angle η subtended at the satellite by δ and the image plane azimuth ψ measured from the heading line is illustrated in Figure 30. The transformation is

$$\tan\eta = \frac{R\sin\delta}{R(1-\cos\delta) + H)}$$

$$\psi = \alpha - \alpha^* .$$

As illustrated in Figure 31, coordinates (η,ψ) transform to Cartesian coordinates in the image plane according to

$$x = (f\tan\eta)\sin\psi$$

$$y = (f\tan\eta)\cos\psi .$$

In these equations f is a scale factor related to the field of view of the imaging device. The y axis is along the heading line and, as mentioned above, the origin of coordinates is at the image center (the image of the subpoint).

The equations that connect the UTM system with latitude and longitude are

$$E = \alpha\sin^{-1}[\sin(\lambda - cm)\cos\phi]$$

$$N = \alpha\sin^{-1}\left[\frac{\sin\phi}{\cos(E/\alpha)}\right]$$

where E is the easting coordinate, N is the northing coordinate, $\alpha = 0.9996R$, and cm is the longitude of the central meridian of the UTM zone. These equations are also specialized to a spherical earth, and do not show the false easting (and false northing in the Southern Hemisphere) that must be added.

A large effect on the geometry of the satellite image is the rotation of the earth. The earth's rotation causes the heading of the ground track to

Figure 29. Transformation from Latitude-Longitude to Great Circle-Azimuth. [7]

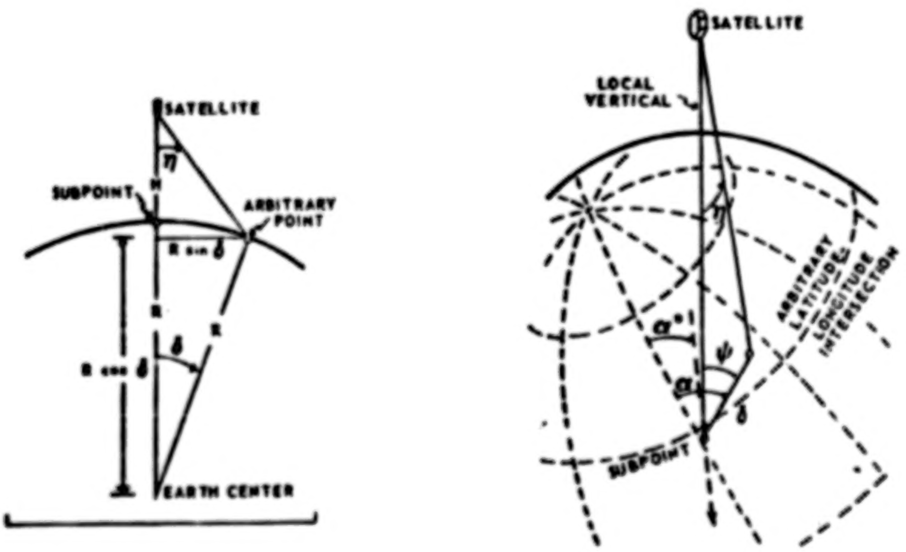


Figure 30. Transformation from Great Circle-Azimuth to Image Nadir-Azimuth. [7]

deviate, and produces skew in the image. The earth's rotation causes the plane viewed to move eastward, so successive points along the ground track are farther and farther to the west of where they would be in the absence of earth rotation. During the scan period of 33 msec., a point at the equator is displaced due to earth rotation by approximately 15 meters. During the 25 seconds required to scan a frame, the shift at the equator is 13,500 meters, and decreases as the distance from the point on the earth's surface to the earth's axis, which is proportional to the cosine of the latitude. Referring to Figure 32 [8] the dotted meridian of longitude through S rotates to point S_o which is a subsatellite point at position given by latitude Φ, longitude λ_s. Thus the satellite views the point S on the earth's surface, located a distance Δ from the point S_o which would have been viewed in the absence of earth rotation. The effective satellite track is that through S, which has heading H greater than the original heading H_o. The distance Δ varies as the angular velocity V_E of the earth and the cosine of the latitude, and the distance Δx covered by the satellite during the same period of time varies as the velocity V_s of the satellite. Hence

$$\frac{\Delta}{\Delta x} = \frac{V_E}{V_s} \cos \Phi$$

where $V_E/V_s = 0.0717$ for Landsat -1.

In the MSS image (Figure 33), point S on the earth's surface appears at point S_o due to west-to-east movement of the earth's surface by a distance Δ while the satellite covers a distance Δx, and hence is scanning along the line through S_o. Lines of latitude are rotated by the local heading angle, H_o, plus an additional amount dH due to the earth's rotation. The change in position Δ has components dx (along satellite motion) and dy (along scan directions). In the practical case, for small heading angle, dx is small and the effect is that successive scan lines cover a portion of the earth farther and farther to the west, and skew is introduced into the resulting image.

The apparent change in heading dH can be determined by writing

$$\tan dH = \frac{\Delta}{OS} = \frac{\frac{\Delta}{\Delta x} \cos H_o}{1 + \frac{\Delta}{\Delta x} \sin H_o} .$$

$$\text{using } OS = \frac{\Delta x + dx}{\cos H_o} \quad \text{and } dx = \Delta \sin H_o .$$

Substituting for $\Delta/\Delta x$ in terms of earth and spacecraft velocities,

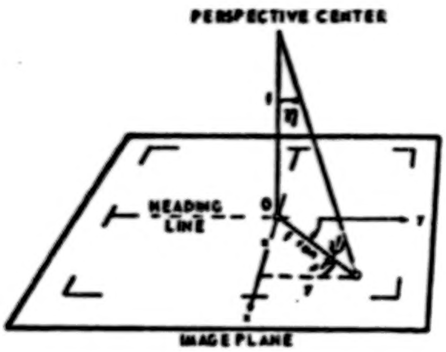


Figure 31. Transformation from Image Nadir-Azimuth to Cartesian Coordinates.

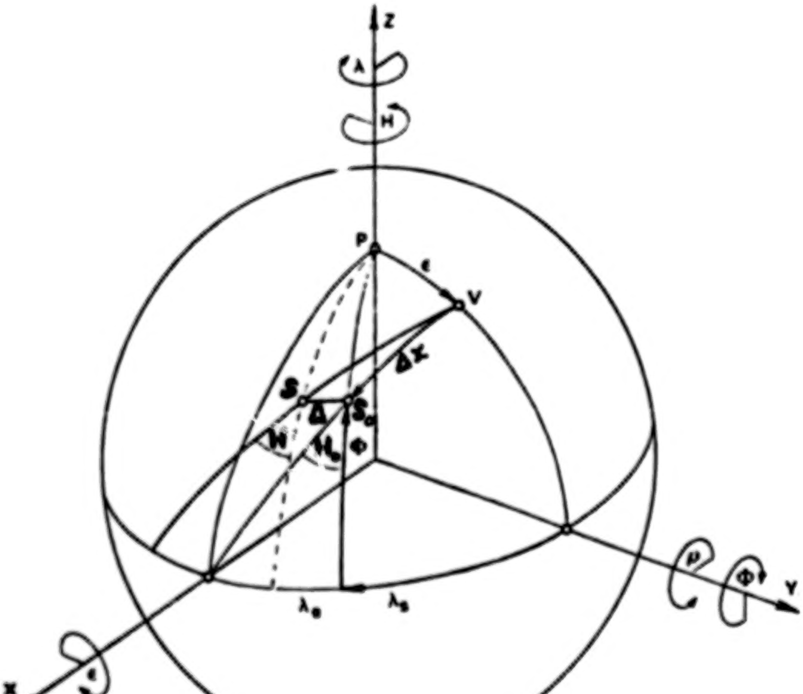


Figure 32. Effect of Earth Rotation on Satellite Track.

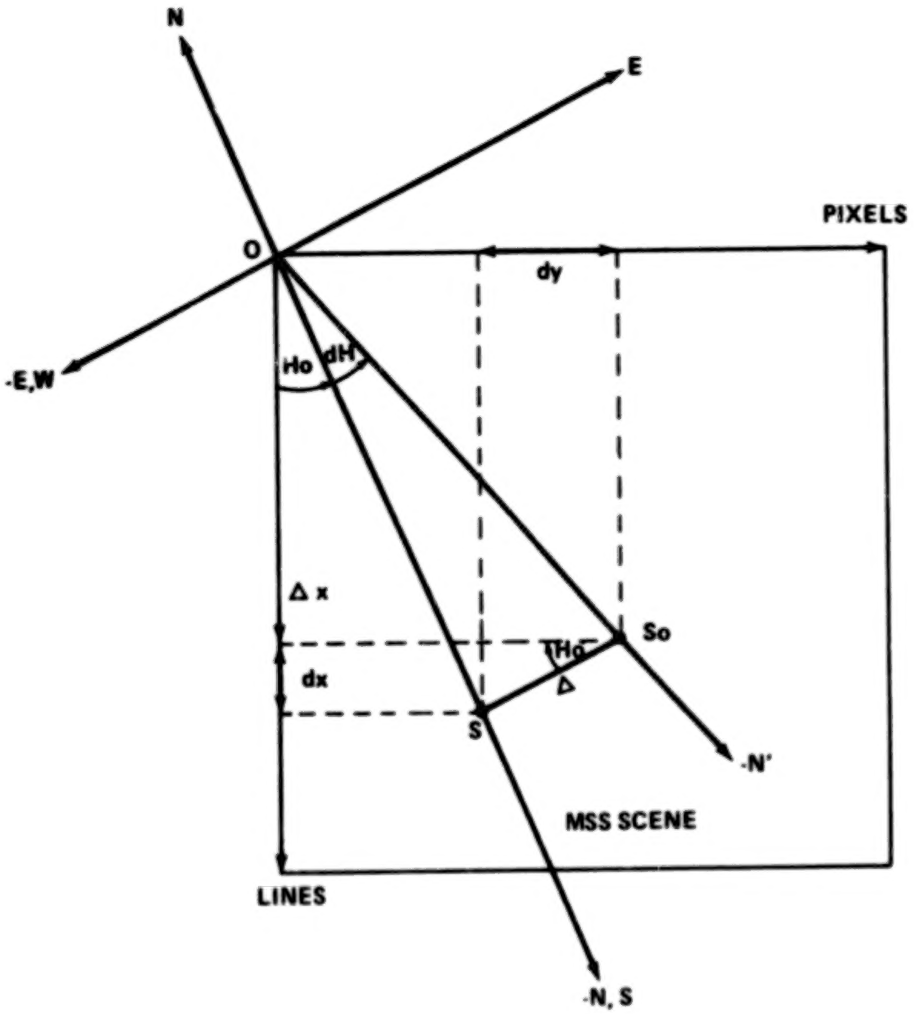


Figure 33. Orientation of UTM Axes in MSS Scene After Rotation and Skew.

$$\tan dH = \frac{\frac{V_E}{V_s} \cos\phi \cos H_o}{1 + \frac{V_E}{V_s} \cos\phi \sin H_o}$$

$$\sim \frac{V_E}{V_s} \cos\phi \cos H_o.$$

For the latitude ($\phi = 34.75°$) of Huntsville, Alabama, the heading H_o with respect to the local meridian is given by

$$H_o = \sin^{-1} \left[\frac{\sin i}{\cos\phi}\right] = 10.86^o$$

using 8.906^o as the polar inclination of the orbit [9]. The skew angle dH is computed to be 3.27^o.

The locations of the scan line and pixel axes in the UTM system on the ground are shown for rotation and skew in Figure 34. The correction for rotation by the heading angle H_o is

$$x' = -E \sin H_o - N \cos H_o$$

$$y' = E \cos H_o - N \sin H_o.$$

The correction for skew, assuming scan lines in an east-west direction, is

$$x'' = x'/\cos dH$$

$$y'' = x' \tan dH + y' = x' \frac{\sin dH}{\cos dH} + y'$$

Substituting for x' and y', we obtain

$$x'' = \frac{-E \sin H_o}{\cos dH} - \frac{N \cos H_o}{\cos dH}$$

$$y'' = -E \left[\sin H_o \frac{\sin dH}{\cos dH} - \cos H_o\right] - N \left[\cos H_o \frac{\sin dH}{\cos dH} + \sin H_o\right].$$

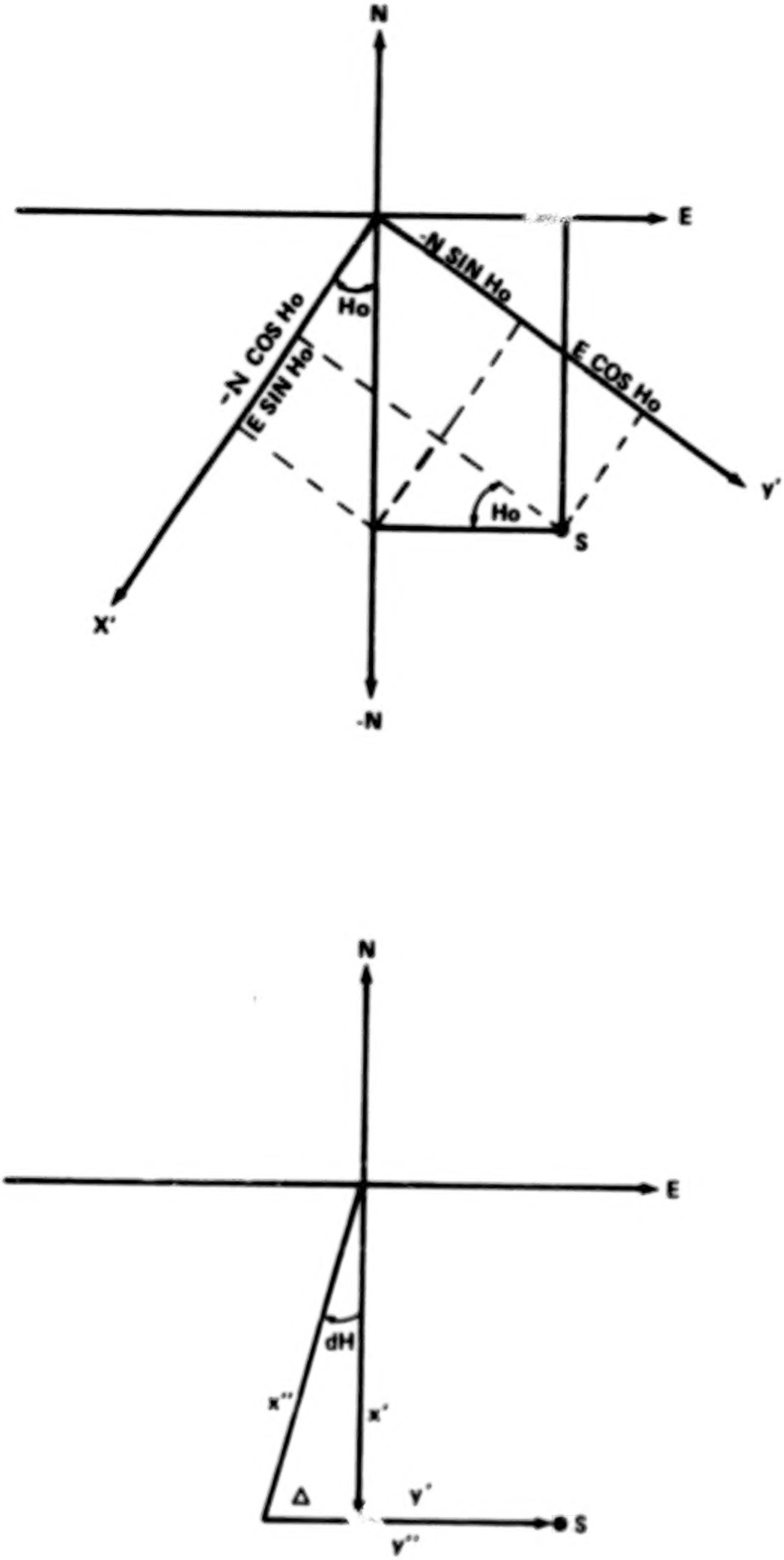


Figure 34. MSS Axes in Terms of UTM Axes for Rotation and Skew.

The transformation from the ground into the image pixel coordinates becomes

$$\begin{bmatrix} x \\ y \end{bmatrix} = \frac{1}{\cos dH} \begin{bmatrix} -\sin H_o & -\cos H_o \\ \cos (H_o + dH) & -\sin (H_o + dH) \end{bmatrix} \begin{bmatrix} E \\ N \end{bmatrix} + \begin{bmatrix} x_o \\ y_o \end{bmatrix}$$

where x is the line count coordinate, y is the scan line point count coordinate, and, from right to left: x_o and y_o are the components of an origin shift vector (so the first point in the first line can have the coordinates $(x, y) = (1, 1)$), E and N are the easting and northing UTM coordinates, H_o is the nominal heading angle, and dH is the skew caused by the earth's rotation. Both the E - N and x-y coordinate systems are right-hand systems; in the image, the line count x increases downward, while the point count coordinate y increases to the right. Although dH does depend on H_o, it is not completely determined by H_o; it also depends on the satellite's angular velocity, the earth's rotational rate, and the latitude, according to the previous expression for tan dH.

The significant thing about the transformation matrix that has been obtained, from the point of view of this discussion, is that for one image the elements of the matrix are (almost exactly) constants. (H_o and dH change due to their latitude dependence, which changes by $1\text{-}2/3^o$ across a scene.) Further, an arbitrary 2 by 2 matrix with constant elements can be assumed to be of the form given in the transformation.

Implicit in the transformation is the assumption that equal distances along a scan line correspond to equal distances on the ground, anywhere along the scan line. This may not actually be so. The multispectral scanner carried on Landsat will be chosen as an example. That scanner is an electromechanical device with scanning performed by a rotating mirror, swinging back and forth (with no imaging performed during the "back" part of the motion). It is clear that, if the angular rate of the mirror is constant, the velocity of the intercept of the line of signt with the ground is not constant. (The combination of this with the forward motion of the spacecraft causes the ground sweep to be slightly S-shaped.) Since the maximum angle of sweep away from the nadir is small, this effect is quite small. In fact, the angular rate is not exactly constant during the sweep. The velocity profile is slightly sinusoidal. The effect of the latter is somewhat greater than that of the former in the case of the Landsat scanner. Other small effects, such as the angular bend due to the change in dH across a scene, are discussed in Reference [10]. Effects such as these can be accounted for by making the matrix elements functions of position. However, the constant-matrix formulation is at least an excellent approximation, so its use in the geographic referencing problem will now be described.

The transformation

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} E \\ N \end{bmatrix} + \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$$

is completely determined if the elements of (a_{ij}) and x_0, y_0 are known. One approach to geographic referencing could be to compute the transformation from orbit parameters and similar information. Then inversion of the transformation, as long as (a_{ij}) is not singular, would give the geographic coordinates corresponding to every point in the picture.

However, this approach has serious weaknesses. Because satellite velocities are in the range 5-10 km/sec, a rather small ephemeris error could cause a significant location error. Also, the approach assumes that there are no attitude, pointing, or motion errors. Those, as well as the nonlinearities discussed in the preceding section, could cause location errors or have the result that values a_{ij} different from those calculated actually characterize the transformation.

There is an approach circumventing these difficulties. The transformation contains six unknown parameters: a_{11}, a_{12}, a_{21}, a_{22}, x_0, y_0. If both sets of coordinates (x, y) and (E, N) are known for at least three points (six components), the unknown parameters can be solved for. If three points are used, the solution is algebraic. If there are more than three points, the resulting set of equations is overdetermined. In this case, a method such as least squares can be used to solve for the six unknowns. The latter approach is preferable; the influence of modeling and observational errors is minimized when a sufficient number of judiciously located "control points" (known points) is used.

Following is an outline of the well-known classical generalized least squares method. Suppose N observations are made of some "observable" y, and y is assumed to have the form

$$y(x) = \sum_{k=1}^{n} a_k f_k(x), \quad n < N \tag{1}$$

The least-squares assumption states that the "best" estimate $\hat{y}$ of y minimizes the function

$$Q = \sum_{i=1}^{N} w_i (\hat{y}_i - y_i)^2 \tag{2}$$

where y_i is the ith observed value of y, $\hat{y}_i$ is the value of (1) at $x = x_i$ with some set of values assigned to $\{a_k\}$, and $\{w_i\}$ is a set of weights. The resulting equation for the set of best estimates of the coefficients a_k is, in matrix notation,

$$\hat{\underline{a}} = [F' \, W \, F]^{-1} \; F' \, W \, \underline{y} \tag{3}$$

Here $\underline{y}$ is the (column) vector whose components are y_i. W is the matrix whose diagonal elements are w_i and whose off-diagonal elements are zero, F is the matrix whose rows are $\underline{f}'$, the transposes (row vectors) of the set of vectors whose elements are $f_k(x_i)$ (one vector for each x_i), and F' is the transpose of F. If y(x) is not a linear function of the fit parameters $\{a_k\}$, the formulation can still be applied. The expression for y(x) is linearized by expanding in a Taylor series and retaining only the leading terms, and then proceeding similarly. Because of the approximation made, the solution is iterative. An equation similar to Equation (3) gives each successive estimate of $\hat{\underline{a}}$, where the right-hand side depends on the result of the previous iteration.

The transformation is linear in the six parameters to be adjusted by the fit, so an iterative formulation is not necessary. However, both the dependent and independent variables are vectors, whereas Equations (1) - (3) only considered scalars. Fortunately, the generalization from one to several dimensions is straightforward. There is essentially no change for the independent variable, which only appears in the equations indirectly, as a summation index labeling points at which observations are made. The dependent variable causes no more trouble. It is perfectly logical to minimize a function of the vector $\vec{\epsilon}_i$ (the deviation between the observed quantity and its estimate at the ith observation), such as

$$Q = \sum_{i=1}^{N} w_i \, \vec{\epsilon}_i \cdot \vec{\epsilon}_i \tag{4}$$

Performing the steps of the analysis shows that Equation (4) treats components of vector observations, for all observations, in the same way that Equation (2) treats scalar observations. Equation (4) is a straightforward generalization of (2) to several dimensions. Another simple change puts the treatment of all components on an equal footing, whether they are the components of the same or different error vectors. It will be noted that, in Equation (4), all components of one vector are given the same weight. This is an unnecessary restriction, and it actually simplifies the mechanization of the equations to remove it and allow each component to have a different weight. Suppose the vectors have M components, labeled by j $(j = 1, 2, \ldots, M)$, and define $m = (i-1)M + j$ (i labels observations, $i = 1, 2, \ldots, N$). Then Equation (4) can be generalized to

$$Q = \sum_{m=1}^{NM} w_m \epsilon_m^2 \tag{5}$$

where the first M terms apply to the first observation, the second M to the second observation, etc. Then a computer program implementing Equation (3) can be used to perform the solution of a multi-dimensional regression problem, with only some changes in indexing.

In order to solve for the parameters of the transformation by the method just described, it is necessary to know the coordinates of several points, called "control points," as accurately as possible in both the UTM and the scan line-point count systems (the latter is called the pixel coordinate system). Control points can be any features that can be readily identified, such as highway intersections, projecting tips of islands or peninsulas, ends of bridges, distinctive buildings, etc. The UTM coordinates of control points can be found quite accurately by reference to standard maps.

Accurate determination of pixel coordinates is more of a problem. For the case of the scanner carried aboard Landsat, for example, an error of a few pixels corresponds to several hundred meters on the ground. Although the least squares fitting may absorb some error, it is unwise to rely on this. (In particular, systematic errors in the same direction will not be removed by the least squares process.) It is highly desirable to locate control points to the nearest pixel. Unless special equipment capable of making highly accurate measurements on small-scale imagery (small enough so that the eye blends pixels together) is available, this requires enlargement sufficient to permit the discrimination of individual pixels. As is discussed below, it is useful to employ some image enhancement techniques in addition to simple enlargement.

The procedures to be described are illustrated in Figures 35-37. Figure 35 shows a portion of (one band of) Landsat frame covering part of North Alabama. The squares marked on it indicate small regions containing distinctive features to be used as control points. The next two figures concentrate on one of those regions (the area around Guntersville, Alabama). Figure 36 shows the result of enlarging the image of the small region by repeating each pixel 15 times in both dimensions. (Approximately the same effect would be produced by photographic enlargement.) It is seen that the blockiness of the image - which makes it easy to count pixels - tends to interfere with the recognition of features. When one is sufficiently far from the picture, the eye smooths out the blockiness; however, then it is impossible to count pixels.

It would appear that smoothing the image would help. Since the effect in general of smoothing is to assign different values to adjacent pixels, where each pixel in the enlarged image corresponds to a fraction of a pixel in

ERTS 1104-15552-6, (1,741) TO (1500,3240) TARCOG REGION

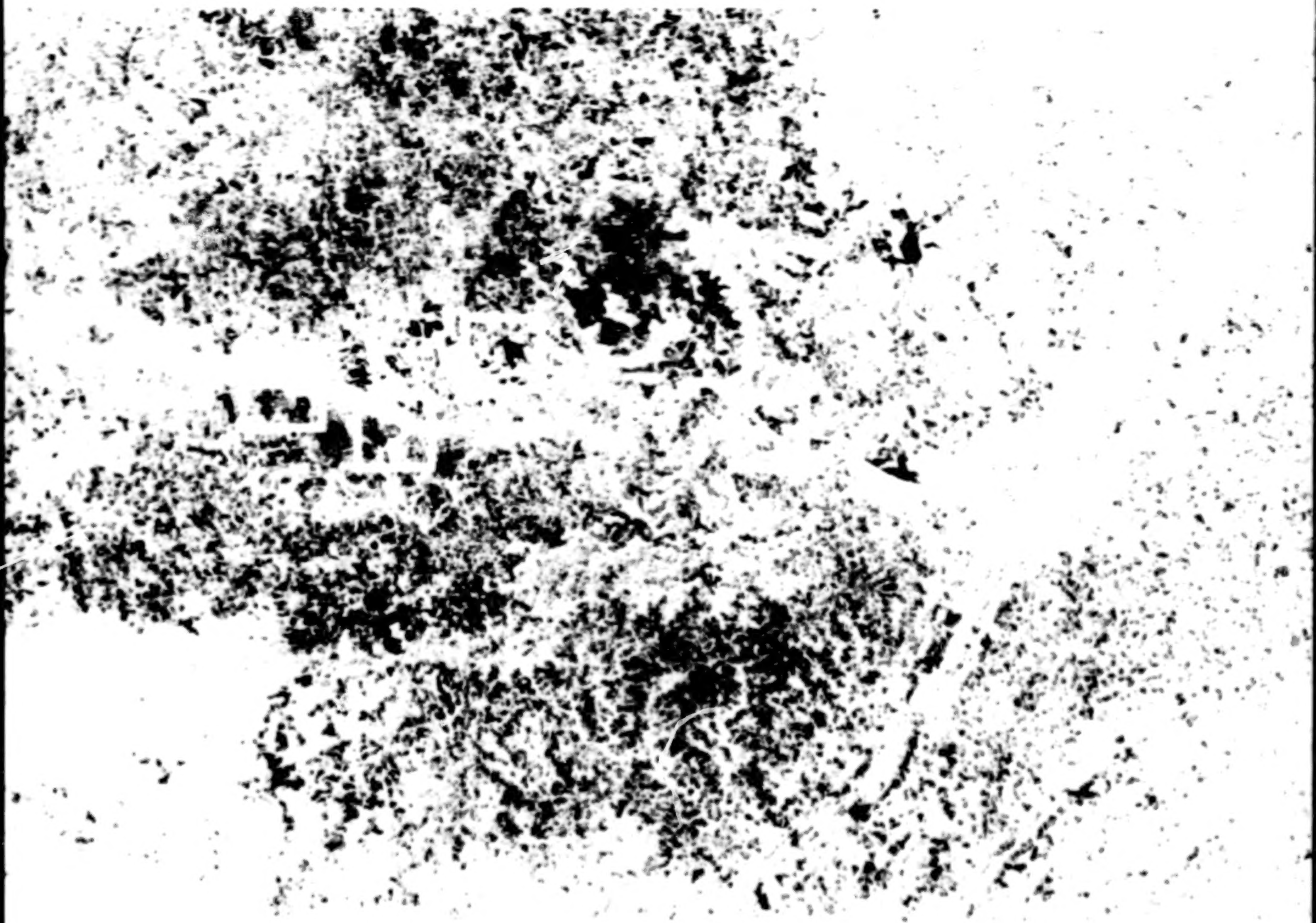


Figure 35. Locations of Ground Control Points for the TARCOG Region.

Figure 36. Enlargement of One Region by Pixel Repetition

the original, smoothing is equivalent to an increase in resolution. The "right" way of achieving this is to use the sampling theorem for interpolation to obtain the intermediate values. The implementation of the appropriate formula is slow computationally, so it is desirable to use approximations requiring less computer time. Figure 37 shows the result of using a bicubic (two-dimensional cubic) interpolation formula approximating the "right" expression to perform the enlargement. In addition, linear density stretching has been applied to effect contrast enhancement.

Working with enlarged imagery, a small measurement error corresponds to only a fraction of a pixel in the original. So it is feasible to locate control points in pixel coordinates to the nearest pixel. Then, with control point locations in both coordinate systems, the least squares solution for the transformation parameters can be performed. It may then be found that the residuals at some control points are excessively large. This situation may persist at a few points even after all errors that can be accounted for have been corrected, due perhaps to errors in the maps used. Such points should be discarded and the solution repeated. (Possibly, subsequent analysis will explain the discrepancy.) The solution is relatively insensitive to the number of points used, unless that number is close to the minimum.

It is appropriate to mention here that these methods may only need to be used the first time geographic referencing is applied to a scene. For other observations of the same scene - for example, subsequent Landsat passes - it may be possible to avoid repeating all of the same steps. Instead, small regions surrounding the control points (whose locations are known) from the first image can be used as templates to search for the locations (pixel coordinates) of the control points in the other observations. Fast sequential methods for doing so exist. [11, 12]

Figure 37. Enlargement of One Region Using Bicubic Interpolation, Enhanced by Linear Density Stretching

3-4. GEOMETRIC CORRECTION

The transformation from ground or UTM coordinates to image pixel coordinates may be used to determine the pixels required in constructing an image in conformance to a UTM map projection. For every UTM grid position (typically at 50 meter spacing) the corresponding pixel coordinates are calculated, and the density at that point becomes the output pixel density. In general, the calculated pixel coordinates are not integers, i.e. the location is between image pixels. Hence, the density must be interpolated from the neighboring pixels. Three interpolation methods will be presented.

For nearest neighbor resampling, the pixel value closest to the position of the correct image pixel is chosen for the result of the interpolation operation. In other words, the coordinates (x',y') of the desired pixel are computed by rounding off the computed coordinates (x,y) to the nearest integer, using

$$x' = x + 0.5$$

$$y' = y + 0.5.$$

This leads to a position error in the nearest neighbor resampled image as large as $\pm$ 0.5 pixel spacing. However, an advantage is that the magnitudes of the samples are retained exactly.

The bilinear interpolation method scales the output value linearly between the density values of two neighboring pixels. If the neighboring pixels have densities A and B, then the scaled density at a distance Δx from A is

$$Q_1 = A + \Delta x\,(B - A).$$

In two dimensions, the input values are the four corners of the square containing the calculated pixel location. If A and B are the densities of the top two corners and C, D of the bottom, the interpolated output along the bottom line is

$$Q_2 = C + \Delta x\,(D - C).$$

The values Q_1 and Q_2 are then interpolated in the orthogonal direction to give the final result:

$$Q_1 + \Delta y\,(Q_2 - Q_1).$$

For spatial frequency band-limited data, the ideal interpolation function is sin (x)/x. A continuous signal can be sampled at discrete intervals and then the sin (x)/x filter can be applied to the discrete data to completely reconstruct the continuous signal. This can be done provided the sampling frequency meets the Nyquist criterion, i.e. it is at least twice the highest spatial frequency.

However, this function has significant magnitude until very high x, requiring an impractically large number of terms (> 1000) for each interpolated value. In addition, Landsat MSS data is not band-limited, but in fact contains aliasing errors, which are not removable after resampling without severe resolution degradation. Thus a limited extent approximation is made to this function.

The cubic convolution function is an approximation to the sin (x)/x function, maintaining the main positive lobe and the first negative lobe on either side. No term beyond x=2 exists. The functions are shown in Figure 38. In these graphs the x axis can be taken as distance from the resample location to the discrete data locations and the y axis is the response value. The equations of the cubic function for the two lobes may be expressed as

$$f_1(x) = a_1|x^3| + b_1 x^2 + c_1|x| + d_1 \qquad 0 \le |x| \le 1$$

$$f_2(x) = a_2|x^3| + b_2 x^2 + c_2|x| + d_2 \qquad 1 \le |x| \le 2.$$

The eight coefficients may be determined by applying the following eight conditions:

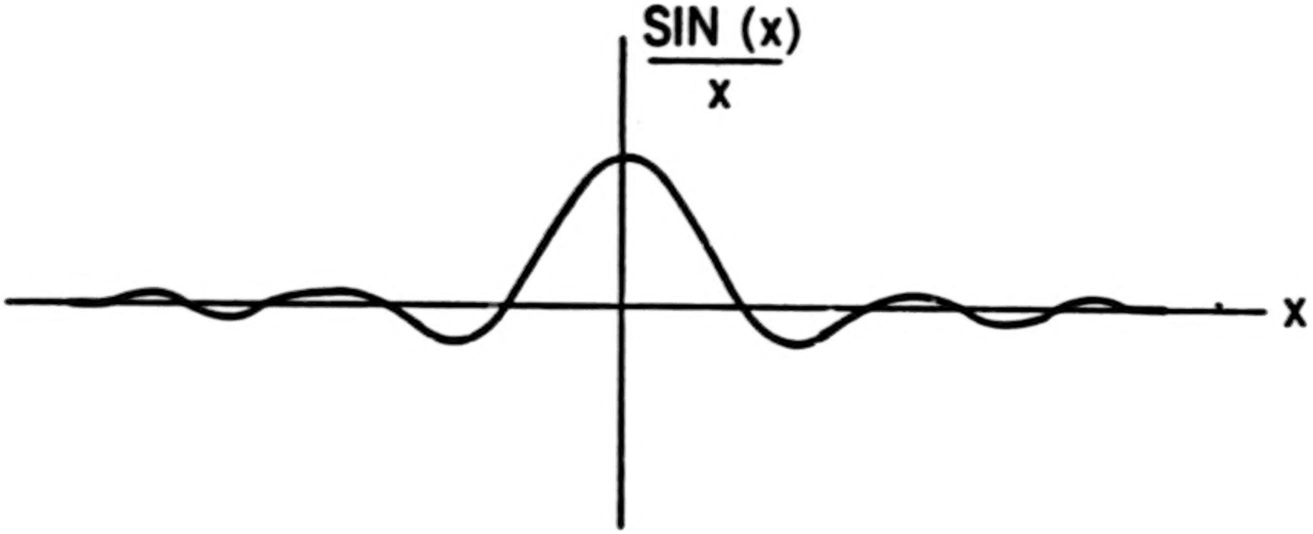
$$f_1(0) = 1$$
$$f_1(1) = 0$$
$$f_2(1) = 0$$
$$f_2(2) = 0$$
$$f_1'(0) = 0$$
$$f_1'(1) = f_2'(1)$$
$$f_1''(0) < 0$$
$$f_2''(1) > 0.$$

The cubic convolution polynomials then become [13]

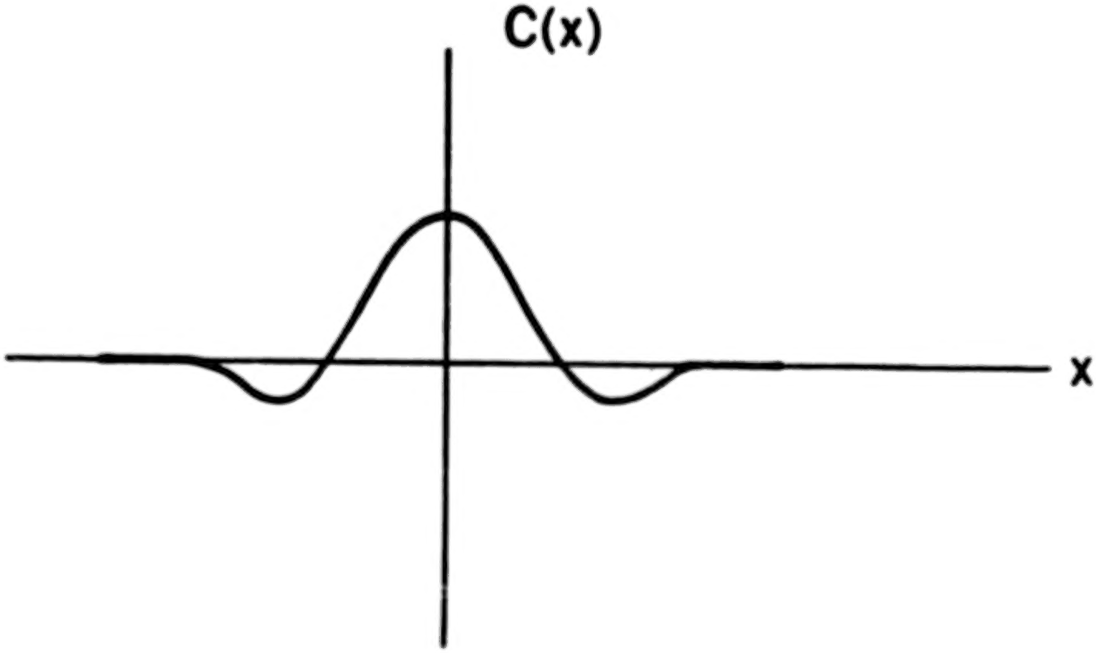
$$f_1(x) = |x|^3 - 2x^2 + 1 \qquad 0 \le |x| \le 1$$

$$f_2(x) = -|x|^3 + 5x^2 - 8|x| + 4 \qquad 1 \le |x| \le 2.$$

Cubic convolution is accomplished using a 4 x 4 pixel subimage about the resample location. First, a vertical axis is passed through the resample location. Next, four horizontal axes are made through the four rows of pixels. At the intersection of the vertical axis and each of the four horizontal axes an interpolation value is computed. Finally, these four interpolated values are reinterpolated along the vertical axis to produce a value at the resample location. The interpolation formula above is used to do each of the five interpolations.



$\frac{\text{SIN (X)}}{\text{X}}$ RESPONSE CURVE



CUBIC CONVOLUTION RESPONSE CURVE

Figure 38. Interpolation Functions for Resampling.

3-5. SUPERPOSITION OF BOUNDARIES

The mathematical details of some of the steps described in Section 2-5 are presented below.

(i). Thinning and Conversion to SLIC

Let P_{ij} be the density at the point (i, j) in the digitized boundary image produced by the microdensitometer. Then, the values p_{ij} are generally available as a sequential file consisting of several records, the ith record consisting of

$$\{p_{ij} \mid j = 1, \ldots, J\} \qquad \text{for } i = 1, \ldots, I.$$

Now, let a threshold t be selected such that all points in the image satisfying $p_{ij} \leq t$ can be decided to be boundary points. The boundary data may then be compressed by setting single bits to "1" at the boundary positions. The boundary lines were thinned by a peeling algorithm which remains outer layers of thick lines while ensuring that connectivities are preserved.

To decide whether a particular boundary point should be deleted (i.e. the bit corresponding to it changed to 0), we examine a 3 x 3 neighborhood centered around the point. Consider the array

$$\begin{array}{ccc} a & b & c \\ d & e & f \\ g & h & i \end{array}$$

where each letter represents a binary pixel. It is to be decided whether e, which is presently equal to 1 should be changed to 0. The conditions for a 'top peel' will be derived below and those for peeling from the other directions follow by symmetry.

First of all, e should be a top boundary point. That is, there should be no boundary point directly above e and there should be a boundary point below e. Therefore b = 0 and h = 1 are necessary conditions. Suppose $\bar{b}h = 1$. (Here, $\bar{b}$ denotes the complement of b). Then, we need only check whether e is a nonessential boundary point, that is, whether two 0's in the 3 x 3 array which are disconnected will stay disconnected where e is made 0. Connectivity, in this context, is defined as the existence of a path not including 1's and consisting only of horizontal and vertical segments.

Now, it is easy to see that e is essential if and only if $a\bar{d} = 1$ or $\bar{c}f = 1$. Therefore, the condition for a top peel is that

$$\bar{b}h\,(\bar{a}+d)\,(\bar{c}+f) = 1.$$

Equivalently, to perform a top peel we set

$$e = e\,(b + \bar{h} + a\bar{d} + c\bar{f}).$$

It is convenient to implement the above equation by employing bit manipulation routines operating on pairs of 32 bit words, thereby performing the top-peel operation in parallel on 32 pixels. This is done by using the "current" array in place of e, the "previous" array for b, the "next" array in place of h. Also, the previous, current, and next arrays are right (left) shifted by one bit and used for a, d and g (c, f and i) respectively in the peeling formulas.

The program minimizes the movement of data in core by using circular buffers for storing the "previous, current and next" arrays. An array J dimensioned 3 is used to store the indices pointing to these arrays (J(1) ⟶ previous, J(2) ⟶ current, J(3) ⟶ next) and after finishing each record, only the array J is updated.

Also, top, left, bottom and right peels are performed one after the other by just one pass through the data (thus minimizing I/O) by storing the intermediate results in core and operating with a phase lag.

(ii). Smoothing

For the boundary data to be useful for extracting interior of regions, it is necessary that the boundary i presented be "contiguous" at all points. Continuity and connectivity in the digital domain can be defined as follows [15].

The points (x_1, y_1) and (x_2, y_2) are said to be 4-adjacent if

$$|x_1 - x_2| + |y_1 - y_2| = 1.$$

The points (x_1, y_1) and (x_2, y_2) are said to be 8-adjacent if

$$\text{Max}\,(|x_1 - x_2|,\ |y_1 - y_2|) = 1$$

A curve is said to be continuous at a point (x, y) on it, if there exists at least one point on the curve which is 8-adjacent to (x, y).

The contiguity count for a point P on a curve is defined as the number of points on the curve that are 8-connected to P.

Two points P and Q on a curve (in a region) are said to be connected if there exists a sequence of points $P_0, P_1, P_2, \ldots, P_n$ such that

$P_0 = P$, $P_n = Q$. P_i is on the curve (in the region) for $i = 0, 1, 2, \ldots, n$ and P_i is 8-adjacent (4-adjacent) to P_{i-1} for $i = 1, 2, \ldots n$.

A curve is said to be closed if, for any point P on it, there exists a sequence of points $P_0 = P, P_1, P_2, \ldots, P_n = P$ on the curve where $n > 1$, P_i and P_{i-1} are 8-adjacent for $i = 1, 2, \ldots, n$.

A region is said to be connected if all points in it are connected to each other.

Now, it is easy to see that closed curves are necessary to separate a given region into several connected subregions. Also, if the contiguity count for every point on a curve is greater than or equal to 2, then the curve is closed.

If closed curves in the continuous domain without retracing (or "burrs") were digitized, then the digital curves would be closed according to the above definition. However, when the boundaries are digitized using a microdensitometer and undergo a thinning process, it is impossible to produce closed boundaries as defined above. But an approximation to closed boundaries can be produced wherein there are closed components which contain the major connected regions of interest and a few "burrs" are retained. Smoothing is the process which converts thinned boundaries into (approximately) closed boundaries.

The smoothing algorithm proceeds as follows. At each point, the contiguity count is determined. This is done by testing the row containing the point and the two adjacent rows to see whether there are any 8-adjacent boundary points. The search is quite simple, if it is remembered that the column coordinates in the SLIC format are arranged in ascending order. Therefore, when looking for boundary points adjacent to (i, j) we need only check the $(i-1)^{st}$ and $(i+1)^{st}$ rows until the column coordinates exceed $j+1$. Also, in the i^{th} row we need only check the column coordinates previous and next to j (assuming no repetitions).

Now, if the contiguity count of a point is less than 2 a neighborhood of the point is examined. The size of the neighborhood determines how large a discontinuity will be patched and should be pre-specified. (A square neighborhood with sides of the order of the thickness of the original, i.e., unthinned, boundaries is generally satisfactory.) Two nearest points (if any) which are not connected either to each other or to the given point are determined. Digital approximations to straight lines joining the given point to these two points are generated and stored as row and column coordinates.

After producing the patches at all points as required, the new boundary points are sorted, merged with the input and arranged in the SLIC format.

(iii) Application of Geometric Transformations

The problem of applying a general geometric transformation on a given boundary image can be stated as follows.

Let $B = \{(i_1, j_1), (i_2, j_2), \ldots, (i_n, j_n)\}$ be a set of points obtained by digitizing a curve using a unit grid in the x-y plane. Let

$$x' = f(x, y)$$
$$y' = g(x, y)$$

be a coordinate transformation. Then, the problem is to find a set of integer coordinates

$$B' = \{(k_1, \ell_1), (k_2, \ell_2), \ldots, (k_m, \ell_m)\}$$

which represent the digitization of the same curve using a unit grid in the x'-y' plane.

This is a resampling problem. It can be solved "exactly" if the original curve in the continuous domain has a bandlimited spectrum and the sampling in the x-y plane is fine enough. In that case, one could reconstruct the curve in the continuous domain using sampling theorem and resample in the x'-y' plane. Since this is a time-consuming process, we use an approximation as follows.

First, generate the set of points $\{(x'_r, y'_r) \quad r = 1, \ldots, n\}$ where

$$x'_r = f(i_r, j_r)$$
$$y'_r = g(i_r, j_r).$$

Now, x'_r and y'_r are, in general, nonintegral. Therefore, we choose the nearest integers to x'_r, y'_r and let them represent the boundary points. Further, to assure that connectivity is preserved after the transformation, we join (x'_r, y'_r) and (x'_s, y'_s) by a straight line whenever (i'_r, j'_r) and (i_s, j_s) are 8-adjacent, and generate a digital approximation to the straight line.

This method can be conveniently implemented with the data in SLIC format (a more convenient format for this particular operation is the "chain code" [1]). The only tricky part of the algorithm is to handle the storage and rearrangement of the coordinates of the new boundary points generated when large images are handled. If the boundary coordinates produced for the entire geometrically transformed image can be held in the main memory at a time, it can be written

out on a sequential file in SLIC format by array sorting in core. Otherwise, it is necessary to dump the coordinate data on a direct access device whenever the core capacity is exceeded and then sort the data from the direct access device.

(iv). Thickening

Boundary lines can be thickened by "growing" each boundary point arount itself by a given amount. This is, if (i, j) is a boundary point, (k, ℓ) is also treated as a boundary point for all (k, ℓ) such that $|i - k| \leq h$ and $|j - \ell| \leq h$. Thickening boundaries in two dimensions starting from the data in SLIC format is accomplished as follows. If $j_1, j_2, \ldots, j_n$ are the column coordinates corresponding to the i^{th} row in the given boundary image, then the set

$$\theta_i = \{j \mid |j - j_r| \leq h \quad \text{for some } r \in [1, n]\}$$

is formed. This represents the horizontally thickened i^{th} row. Now, to thicken in the vertical direction, we simply set the output column coordinate set T_i for the i^{th} row to be

$$T_i = \bigcup_{r=-h}^{h} \theta_{i+r} \, .$$

When T_i is generated, it is arranged in ascending order, repetitions, if any, are eliminated, and the coordinate set is written out as a record on a sequential file.

(v). Generation of Region Identification Maps (RIM)

Starting from the basic definition of connectivity for regions given in Section (ii), we can develop an algorithm to identify separate connected regions given the boundary data. An image consisting of a unique number assigned to each connected region is called a region identification map (RIM). We shall adopt the convention that 0 be used for boundary points and 1 for "exterior" points (i.e., points connected to points in the region outside the rectangle containing the given boundary points). The algorithm to generate a RIM proceeds as follows.

Let $\{b_{ij} \mid i = 1, \ldots, I, \; j = 1, \ldots, J_i\}$ be the set of column coordinates of the boundary points (stored in SLIC format). Choose $N \geq \bar{b} - \underline{b} + 1$ where

$$\overline{b} = \underset{i,j}{\text{Max}}\ b_{ij} \text{ and } \underline{b} = \underset{i,j}{\text{Min}}\ b_{ij}$$

Let p and q be two N-vectors. These are the vectors in which the region identification numbers (RIN) will be generated. The vector p will be used to store the RIN for the previous row and q will be used to store those for the current row as they are generated. Also a scalar λ is used to count the number of regions found.

Initially, all points in the "previous" row are in the exterior. Therefore, the vector p is initialized with all components equal to unity. Also, λ is set to unity. Now, consider the i^{th} row. The boundary data

$$\{b_{ij} \mid j = 1, 2, \ldots, J_i\}$$

are read from the sequential file. Since in the SLIC format b_{ij} are in ascending order, the points before b_{i1} and after b_{iJ_i} are exterior points. Therefore,

$$q_k = 1 \text{ for } 1 \le k \le b_{i1} - \underline{b} + 1 \text{ and } b_{iJ_i} - \underline{b} + 1 \le k \le N.$$

Also,

$$q_k = 0 \text{ for } k = b_{ij} - \underline{b} + 1,\ j = 1, 2, \ldots, J_i.$$

Now, q_k must be found only for values of k in intervals

$$A_j = \{k \mid b_{ij} - b + 1 < k < b_{i,j+1} - b + 1\} \text{ for } j = 1, 2, \ldots, J_i - 1.$$

For every such interval, there are two possibilities.

Case 1: There is a $k_0 \in A_j$ such that $p_{k_0} \neq 0$. In this case, all points in the interval in the current row are in the region p_{k_0}. Therefore $q_k = p_{k_0}$ for all $k \in A_j$.

Case 2: $p_k = 0$ for all $k \in A_j$. In this case, it is decided that a new region might be beginning. The region count λ is changed to $\lambda+1$. Also, $q_k = \lambda$ for all $k \in A_j$.

Now, q contains the RIN for the current row. The array q can be written out and also moved into array p, to make it the "previous" row for handling the $(i+1)^{st}$ row.

This procedure, as described so far, produces RIN's assuring that no two unconnected regions will have the same RIN. However, in many cases, the same region may have more than one RIN. This happens since connectivity in the $(i+1)^{st}$ through I^{th} rows is not known when the i^{th} row of the RIM is being generated. Therefore, it is necessary to update the region numbers after connectivity between differently numbered regions is discovered. To do this, a "Region Identity Matrix (RIMX)" D is used to store the information about the connectivity between differently numbered regions. The matrix D is a binary matrix with $d_{ij} = 1$ if regions numbered i and j are connected and 0 otherwise. Initially, D is set equal to a null matrix. When a new region number λ is started, $d_{\lambda\lambda}$ is set to 1. Also, after the vector q is found for the i^{th} row, D is modified by letting

$$d_{\ell k} = d_{k\ell} = 1 \text{ for all } (k, \ell) \text{ such that } k = p_j \neq 0,\ \ell = q_j \neq 0,\ j = 1, \ldots, N.$$

Now, at any stage, the matrix D indicates which region numbers determined thus far represent the same region. This is analogous to the connectivity matrix used widely in graph theory [16].

Connectivity matrices have some interesting properties which are very useful in this application. These will be introduced briefly here. Let λ and μ be two region numbers. Suppose there exists a sequence of region numbers $\lambda_0, \lambda_1, \lambda_2, \ldots, \lambda_n$ such that $\lambda = \lambda_0, \mu = \lambda_n$ and $d_{\lambda_i \lambda_{i+1}} = 1$ for $i = 0, 1, \ldots, n-1$. Then the regions λ and μ are said to be connected by a path of length n. Now, if D_n is evaluated using ordinary matrix multiplication, then the $(\lambda, \mu)^{th}$ element will be equal to the number of paths of length n between λ and μ. Instead, if a logical matrix product is used (using $1+1 = 1$, $1+0 = 1$, $1 \times 1 = 1$ and $1 \times 0 = 0$) find D^n, then the $(\lambda, \mu)^{th}$ element of D will indicate whether region μ can be reached from λ via a path of length n. If the matrix R is defined as

$$R = D + D^2 + D^3 + \ldots + D^n$$

where n is chosen such that $D^{n+1} = D^n$, then $R_\lambda = 1$ if there exists a path between λ and μ of any length and $R_\lambda = 0$ otherwise.

An efficient method to obtain R is to generate a sequence of matrices recursively:

$$R^0 = D$$

$$R^i = R^{(i-1)} + (R^i \times R^i) \qquad \text{for } i = 1, 2, 3, \ldots .$$

The computations are stopped when $R^n = R^{(n-1)}$. (The matrix R^i then indicates paths of length less than or equal to 2^i.)

Now, the matrix R can be used to find the smallest RIN to be assigned to each connected region to which several RIN's have been given. The records of the RIM computed can then be updated using table lookup.

When handling large images, it might become necessary to perform several such updates, depending on the memory assigned to the computation of D and R. If the size assigned to D is exceeded during the computation of the i^{th} row, the (i-1) rows before that are updated using the corresponding matrix R, the updated $(i-1)^{st}$ row is stored in p, the value of λ is set to the largest region number in the updated (i-1) rows of the RIM, D is set equal to an identity matrix and the computation for the i^{th} row is restarted.

Several steps are involved in superposing political boundaries on remotely sensed images. The complexity of handling this problem depends on the facilities available for digitizing the boundary information. The steps described in this memorandum have been designed to handle data digitized using a microdensitometer. The process is considerably simplified if a digitizing plotter/tracer is used so that the boundaries can be digitized by manually tracing the curves from standard maps. In that case, each region can be digitized separately as indicated in [16]. Converting the data corresponding to each region after geometric correction into the so-called "Tightly Closed Boundary" (TCB) format, wherein the extrema and inflections of the boundary are repeated, we would then have a very simple method for extracting individual regions or generating an RIM.

IV. RESULTS

4-1. PRELIMINARY DATA HANDLING

Landsat coverage of the TARCOG region was extracted from the computer compatible tape of scene 1104-15552, obtained on November 4, 1972. The region extracted was lines 1 to 741 and samples 1500 to 3240. (Sample 3240 is the last sample in the scene, due to the fact that the TARCOG region extends out of the Landsat scene slightly.)

4-2. COMPUTER CLASSIFICATION RESULTS

Training samples were selected from a region of size 500 x 500 pixels centered on the city of Huntsville. Two sets of training samples were selected. One set was chosen to be representative of four major land use classes (urban, agriculture, forest, and water); and the other of seven Level I land use classes (urban and built-up, agriculture, forest, wetland, pasture, water, and barren). The training areas were shown in Figure 14.

Linear decision functions were then computed using these sets of training data. The coefficients of the decision functions, in the order in which testing for a positive result is performed, are given in Tables 1 and 2.

The decision functions are then tested by using them to classify the training data. This procedure gives a measure of the accuracy of the decision functions in classifying the training data, but is no guarantee of the results when applied to unknown data from other parts of the scene. This is because there may be present data corresponding to a certain class, but differing sufficiently from the training data of that class that it is classified incorrectly. This situation arises when the training data is not representative of all data corresponding to each class type. However, the separation of the training sample data by the discriminant functions is accurate to approximately 95 percent. The classification assignments of the training data are given in Tables 3 and 4. Using an IBM 360/65 computer, the computer time required to calculate the four class discriminants was 39 seconds. For the seven class discriminants the CPU time was 75 seconds. In each case, 100 training samples for each class were used in the calculations.

The discriminant functions were then tested on the 500 x 500 pixel Huntsville scene, since the land usage of this relatively small area was well known. The class occupancy of this area by number of samples and percentages is given in Tables 5 and 6.

A classification map showing seven classes in the Huntsville region is given in Figure 39.

The classification into four classes required 21 minutes, 18 seconds of computer time. The rate of classification is 2933 pixels per second or 0.3409 millisec. per pixel.

For seven classes, the corresponding values are 1869 pixels per second or 0.5350 millisec. per pixel.

For comparison purposes, Figures 40 and 41 show land use maps of the Jetport region obtained by computer analysis of high altitude (60,000 ft.) three

band photography and by manual analysis of low altitude (6,000-12,000 ft.) four band photography. It is apparent that the areas of significant sizes are classified into the same land usage in each case.

Table 1. Four Class Linear Discriminant Coefficients

(In order of testing; discriminant function is

$$G = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + w_5)$$

Land Use Class	Coefficients				
	w_1	w_2	w_3	w_4	w_5
Water	0.1100	-0.1160	0.07007	-0.3798	0.9280
Forest	-0.2492	-0.1186	-0.1635	0.2078	7.895
Agriculture	-0.4470	0.09002	-0.1091	0.2744	7.973
Urban	0.4470	-0.09002	-0.1091	-0.2744	-7.973

Table 2. Seven Class Linear Discriminant Coefficients

Land Use Class	Coefficients				
Barren	0.07296	0.04997	-0.09904	0.1298	- 3.004
Pasture	0.06251	-0.1843	0.06783	0.1214	- 2.784
Water	0.2855	-0.06975	-0.3331	0.2061	- 2.290
Wetland	0.01383	0.09687	-0.1588	-0.3112	2.905
Urban	0.5952	-0.3246	0.2419	-0.2677	-10.44
Forest	-0.4142	-0.2501	-0.1709	0.4392	10.35
Agriculture	0.4142	0.2501	0.1709	-0.4392	-10.35

Table 3. Classification of 4 Class Training Samples

Land Use Class	Percent Correct	Number of Samples Classified as:			
		Urban	Agriculture	Forest	Water
Urban	95	95	5	0	0
Agriculture	97	2	97	1	0
Forest	98	0	1	98	1
Water	100	0	0	3	100

Average Accuracy = 97.5%

Table 4. Classification of 7 Class Training Samples

Land Use Class	Percent Correct	Number of Samples Classified as:						
		Urban	Agriculture	Forest	Wetland	Pasture	Water	Barren
Urban	82	82	0	0	0	0	0	17
Agriculture	99	0	99	1	0	0	0	0
Forest	99	0	0	99	1	0	0	0
Wetland	97	0	0	1	97	0	2	0
Pasture	99	0	0	1	0	99	0	0
Water	100	0	0	0	0	0	98	0
Barren	87	13	0	0	0	0	0	87

Average Accuracy = 94.7%

Table 5. Four Class TARCOG Land Use

Class	Number of Samples	Percentage
Urban	241,445	6.44
Agriculture	1,378,609	36.76
Forest	2,003,588	53.43
Water	126,358	3.37
TOTAL	3,750,000	

Table 6. Seven Class TARCOG Land Use

Class	Number of Samples	Percentage
Urban	303,545	8.09
Agriculture	969,926	25.86
Forest	2,021,475	53.91
Wetland	38,745	1.03
Pasture	321,830	8.58
Water	92,104	2.46
Barren	2,375	0.06
TOTAL	3,750,000	

Figure 29. Seven Land Use Classes in the Huntsville Region using Landsat Data.

Urban - Red | Forest - Green | Pasture - Magenta

Agriculture - Yellow | Wetland - Cyan | Water - Blue

Barren - Black

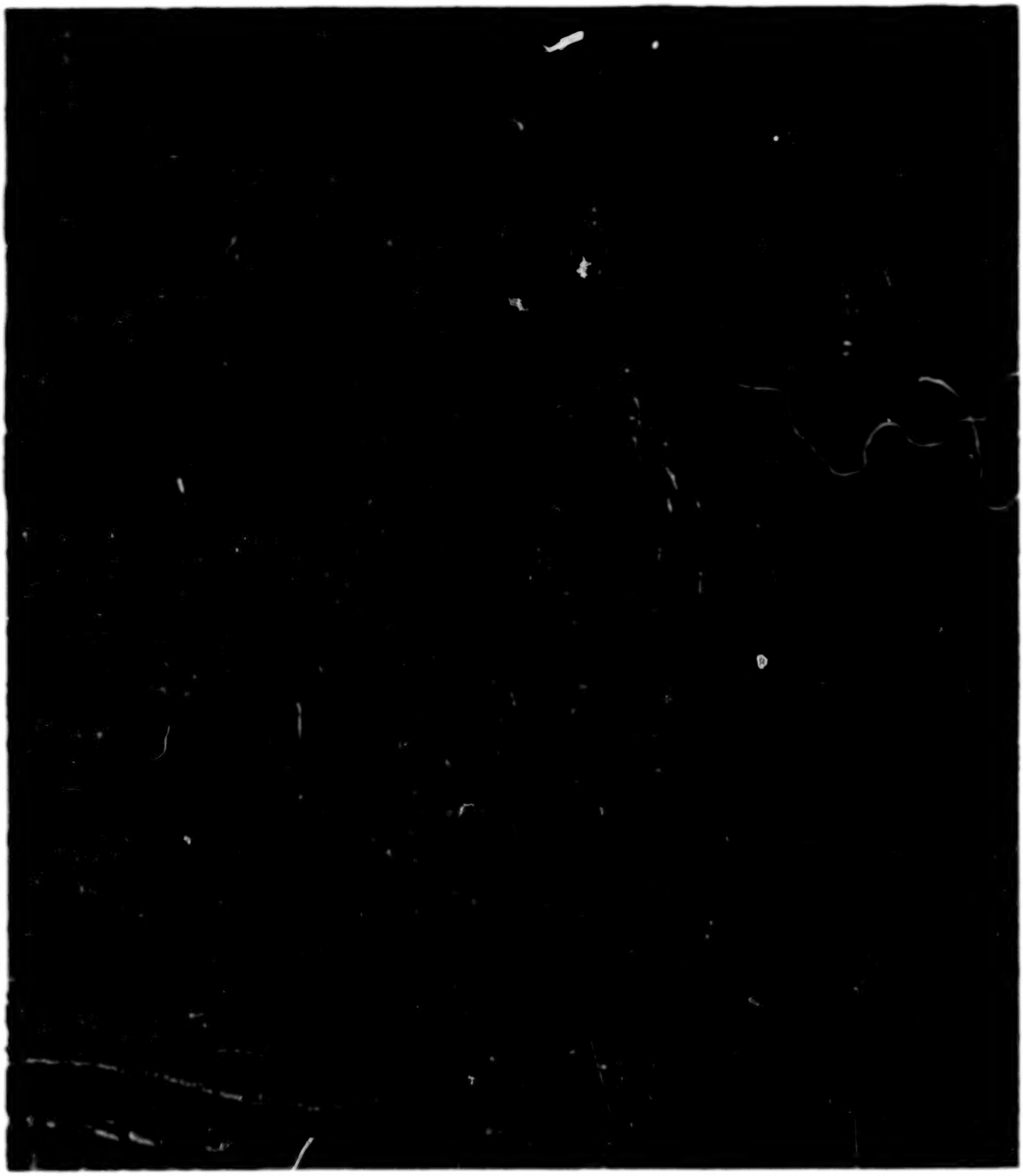


Figure 40. Four Class Map of the Jetport Region, obtained from RB-57 Photography.

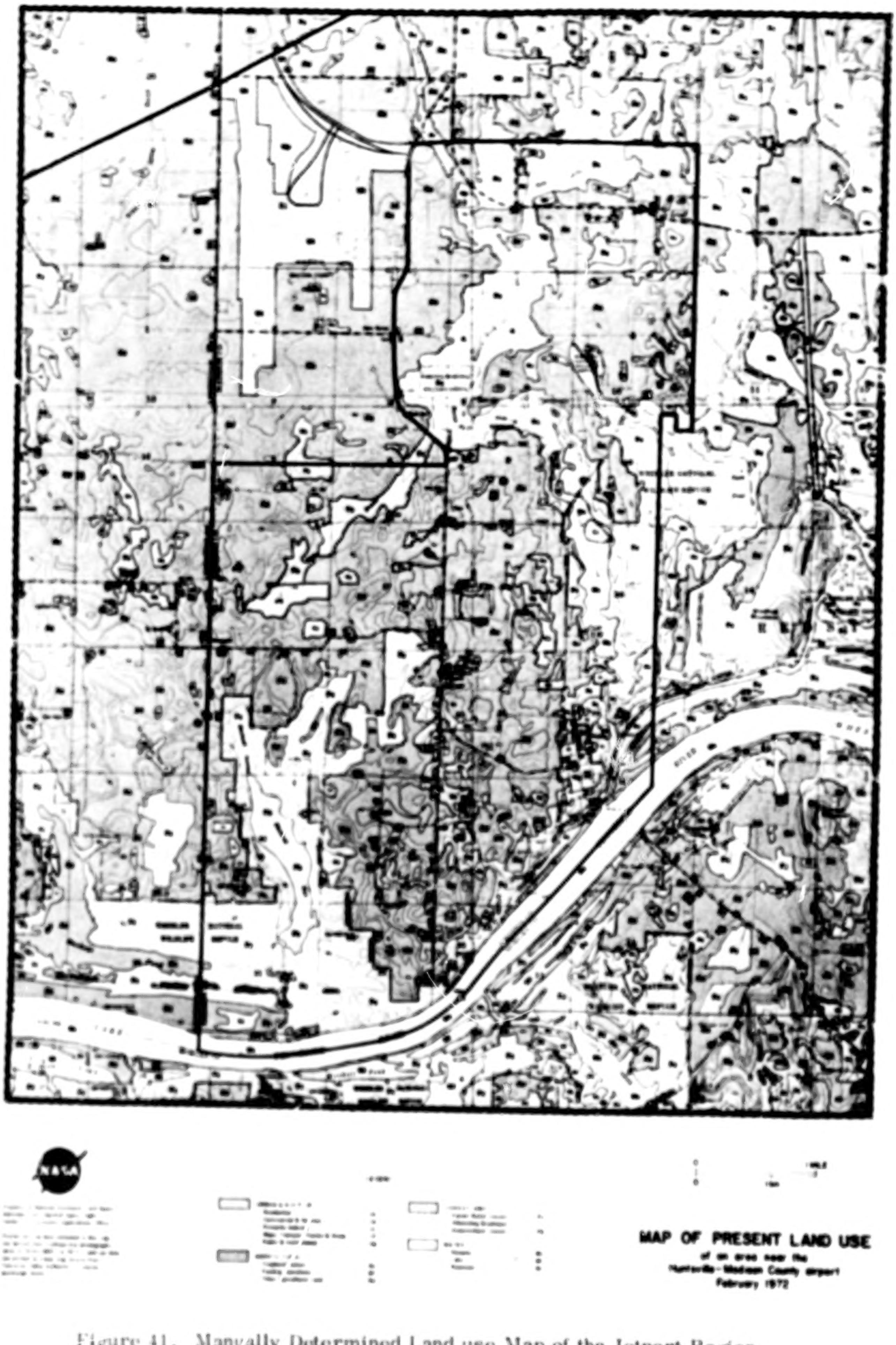


Figure 41. Manually Determined Land use Map of the Jetport Region, obtained from Low Altitude Photography

4-2-1. CLASSIFICATION ACCURACY

The accuracy of the classification has been studied in detail. In one procedure, a ground truth study was conducted in 101 randomly selected study areas located in Madison County. [17] Each study area consisted of a five pixel by five pixel matrix, centered on the random location. Thus the 101 study areas resulted in 2525 pixels, the locations of which were visited in the field, classified and compared with the computer designations. It was determined that 67.4 percent of the study pixels were correctly identified. Since agriculture and pasture are the same in Level I classification, these two groups can be combined. When this is done, the percentage of correctly identified pixels rises to 76.3 percent. Table 7 gives the classifications of the 2525 pixels whose actual land use was determined. From Table 8 it is seen that for a pixel classified as urban there is a 0.675 probability that it is actually urban, a 0.134 probability that it is agriculture/pasture, a 0.025 probability that it is water, and a 0.162 probability that it is actually water. The Bayesian probability of a pixel being classified correctly is the probability of correct classification divided by the sum of the probabilities of other actual classes being so classified. The Bayesian probabilities of correct classification in each land use category are:

urban	0.677
agriculture/pasture	0.465
forest	0.618
water	0.992
wetland	0.519

A second accuracy analysis was performed by examining low altitude photographs of rural areas on Sand Mountain, since it is known that bare soil in agricultural areas is easily confused with urban areas. This is illustrated in Figure 42, in which the outlined areas appear as urban, since they are bright in the green band image, but are in fact are agricultural land usage, as determined from the low altitude photography. In the classification map, light areas are urban, gray agriculture, and dark are forest.

Table 7. Classification of Actual Land Use Classes [17]

Actual Land Use Class	Number of Pixels	Percent Classified Correctly	Actual Land Use Pixels Classified as: Urban	Agriculture & Pasture	Forest	Water	Wetland	Barren
Urban	126	67.5	85	35	4	0	1	1
Agriculture & Pasture	1444	72.8	193	1051	194	5	0	1
Forest	906	85.2	23	108	772	0	3	0
Water	37	40.5	6	4	1	15	11	0
Wetland	12	33.3	0	4	4	0	4	0
Barren	0	0	0	0	0	0	0	0

Table 8. Classification Probabilities of Actual Land Use Pixels

Actual Land Use Class	Number of Pixels	Probability of Actual Land Use Pixels Classified as:				
		Urban	Agriculture & Pasture	Forest	Water	Wetland
Urban	126	.675	.278	.032		.008
Agriculture & Pasture	1444	.134	.728	.134	.003	
Forest	906	.025	.119	.852		.003
Water	37	.162	.108	.027	.405	.297
Wetland	12		.333	.333		.333

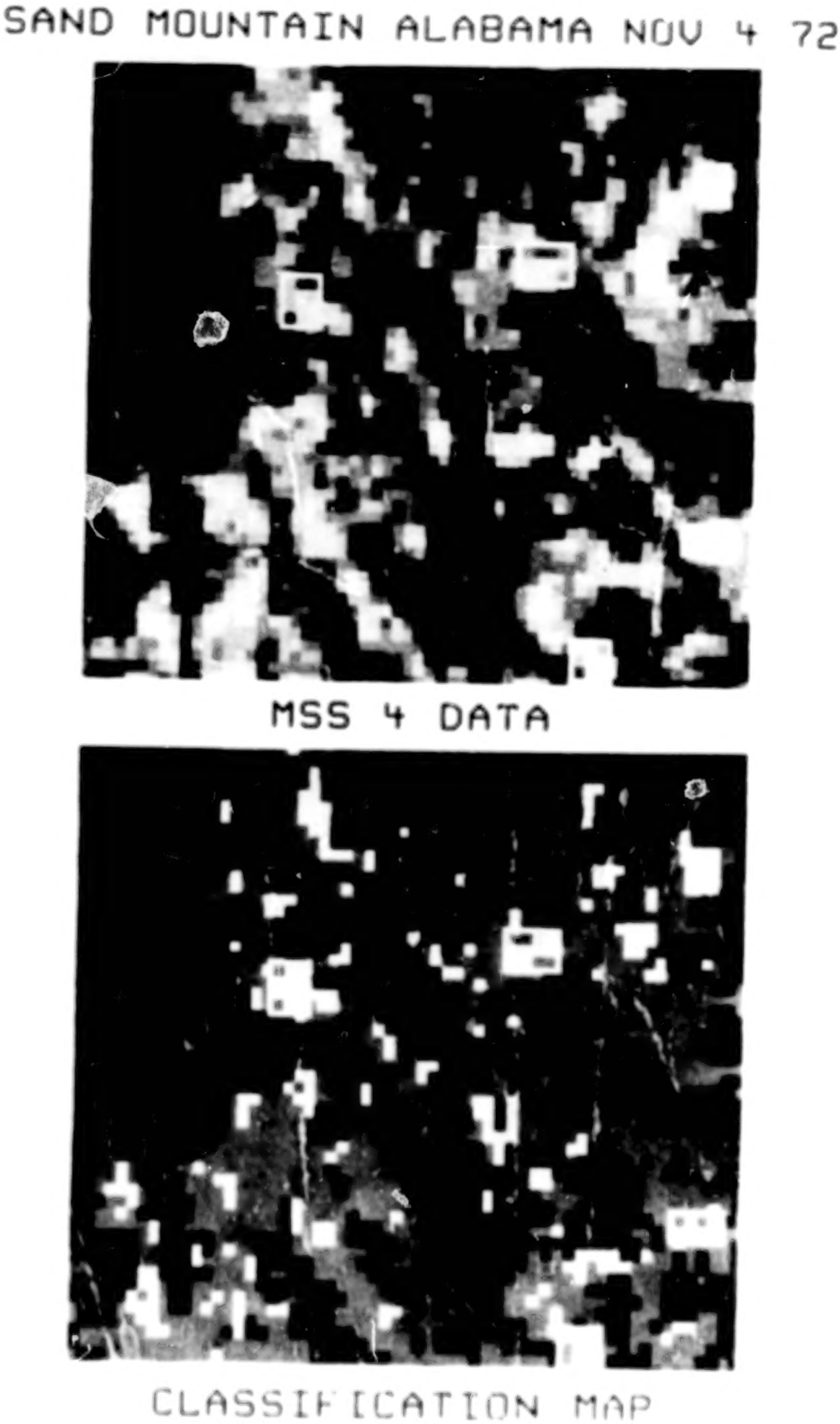


Figure 42. Example of Agriculture Misclassification on Sand Mountain.

Table of Contents

Table of Contents (Cont'd.)

LIST OF ILLUSTRATIONS

Figure

LIST OF ILLUSTRATIONS (Cont'd.)

Figure

LIST OF TABLES

4-2-2. POPULATION DENSITY OF URBAN AREAS

The area assigned to the urban and built-up class for cities in the TARCOG region should bear a constant ratio to their populations, insofar as the type of housing and the proportion of commercial and industrial development remain constant. The built-up areas of eleven cities in the TARCOG region were determined by counting the pixels assigned to the urban category within a rectangular region encompassing each city. The populations used were the published values for the 1970 census. The populations, areas, and population densities are given in Table 9. The pixel counts given were obtained from a geometrically corrected image in which each pixel represented an area of 57 m x 57 m. The populations vs. area are shown plotted in Figure 43. The solid line in the figure is a least squares fit of a linear function. The equation of fit is

$$p = 1312.8\,A - 974.4$$

where p is the population and A is the area in square kilometers.

A previous study [14] of forty urban areas in the Tennessee River Valley using aerial photography yielded the following fit equations for the years 1953 and 1963, respectively:

$$p = 1778.3\,A - 549.4$$
$$p = 1118.7\,A - 2928.4$$

The slope of the fit curve reported in this study falls between these two values. Thus it appears that reasonably consistent results are obtained using computer classified satellite imagery and manually interpreted aerial photography. The average ratio obtained from the three fits is 1403 persons per square kilometer.

Table 9. Population Data For TARCOG Cities

City	Population	Area		Population Density Pop./km^2
		pixels ($57\ m^2$)	km^2	
Huntsville	137802	31267	101.59	1356.5
Decatur	41800	13273	43.12	969.3
Athens	14360	3583	11.64	1233.6
Cullman	12900	2457	7.98	1616.0
Albertville	9963	2485	8.07	1234.0
Scottsboro	9324	1849	6.01	1552.1
Hartselle	7355	2192	7.12	1032.7
Guntersville	6491	902	2.93	2214.9
Boaz	5621	1551	5.04	1115.5
Arab	4399	1261	4.10	1073.7
Madison	3086	1032	3.35	920.4

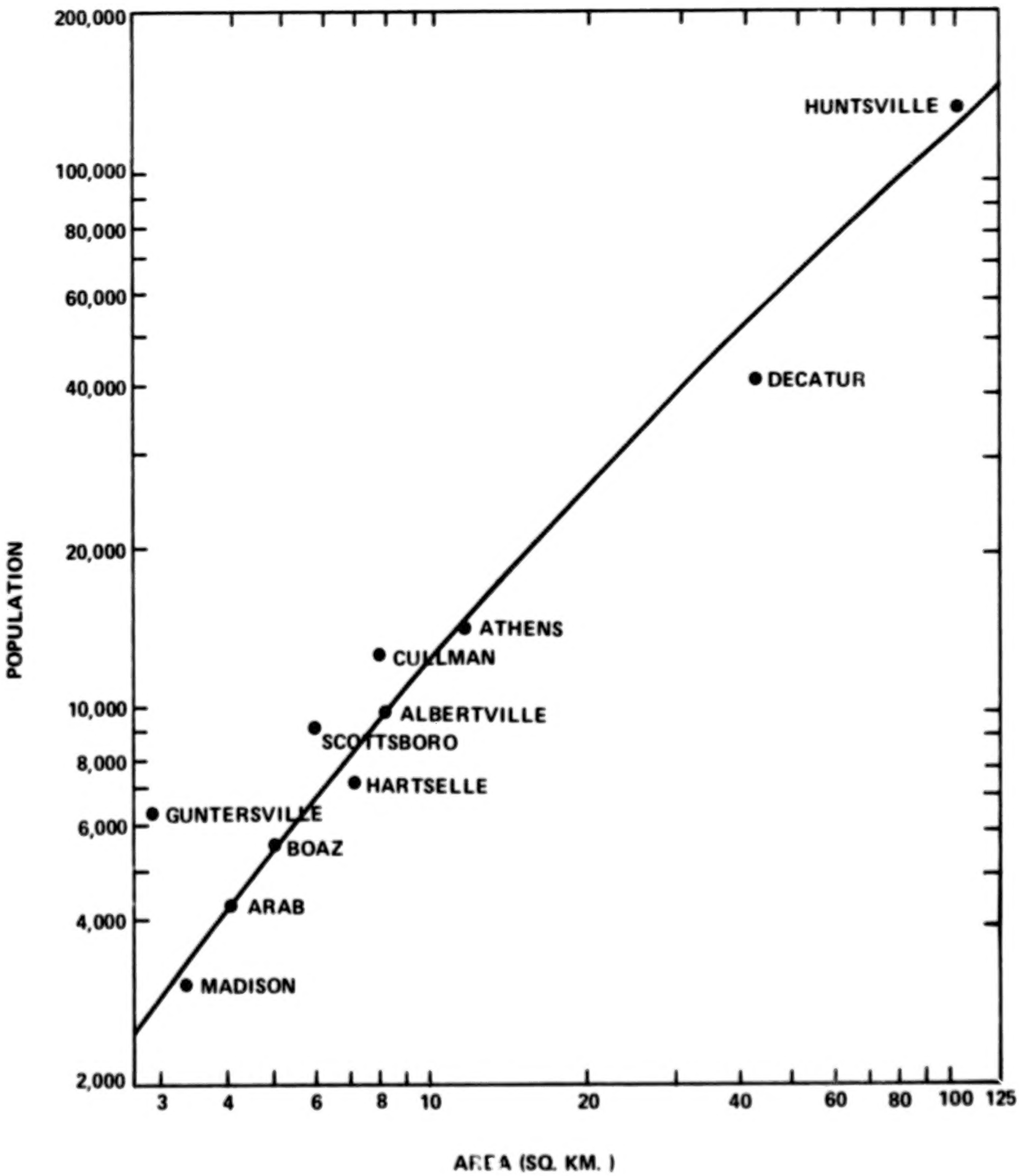


Figure 43. Population Density of TARCOG Cities

4-3. GEOGRAPHIC REFERENCING

Twenty-three control points from the 11 areas marked in Figure 35 were used in the geographic referencing solution. Figure 44 shows a UTM grid superimposed on the region to illustrate the solution. The lines nearly parallel to the sides of the picture run north-south, while the other grid lines run east-west. The spacing between grid lines is 10 km in both directions. The heading, skew, and scale factor distortions are clearly apparent in the figure, as grid squares appear as parallelograms.

The theoretical transformation matrix, considering heading and skew effects, was given previously as

$$\frac{1}{\cos dH}\begin{bmatrix} -\sin H_0 & -\cos H_0 \\ \cos(H_0 + dH) & -\sin(H_0 + dH) \end{bmatrix}$$

where H_0 is the heading angle and dH is the angle of skew.

Evaluating at the center of the scene, latitude 34.5°, the matrix becomes (using $H_0 = 10.83^{\circ}$ and $dH = 3.29^{\circ}$)

$$\begin{bmatrix} -0.18816 & -0.98381 \\ 0.97141 & -0.24423 \end{bmatrix}$$

The scale change between the pixel axes and the UTM axes must be taken into account. The scale in the line count (x) direction is

$$\frac{1}{.079} = 12.66 \text{ pixels/km.}$$

and in the pixel count (y) direction is

$$\frac{1}{.057} = 17.54 \text{ pixels/km.}$$

applying these factors, the matrix becomes

$$\begin{bmatrix} -2.382 & -12.453 \\ 17.042 & -4.285 \end{bmatrix}.$$

The empirical matrix from the least squares fit is

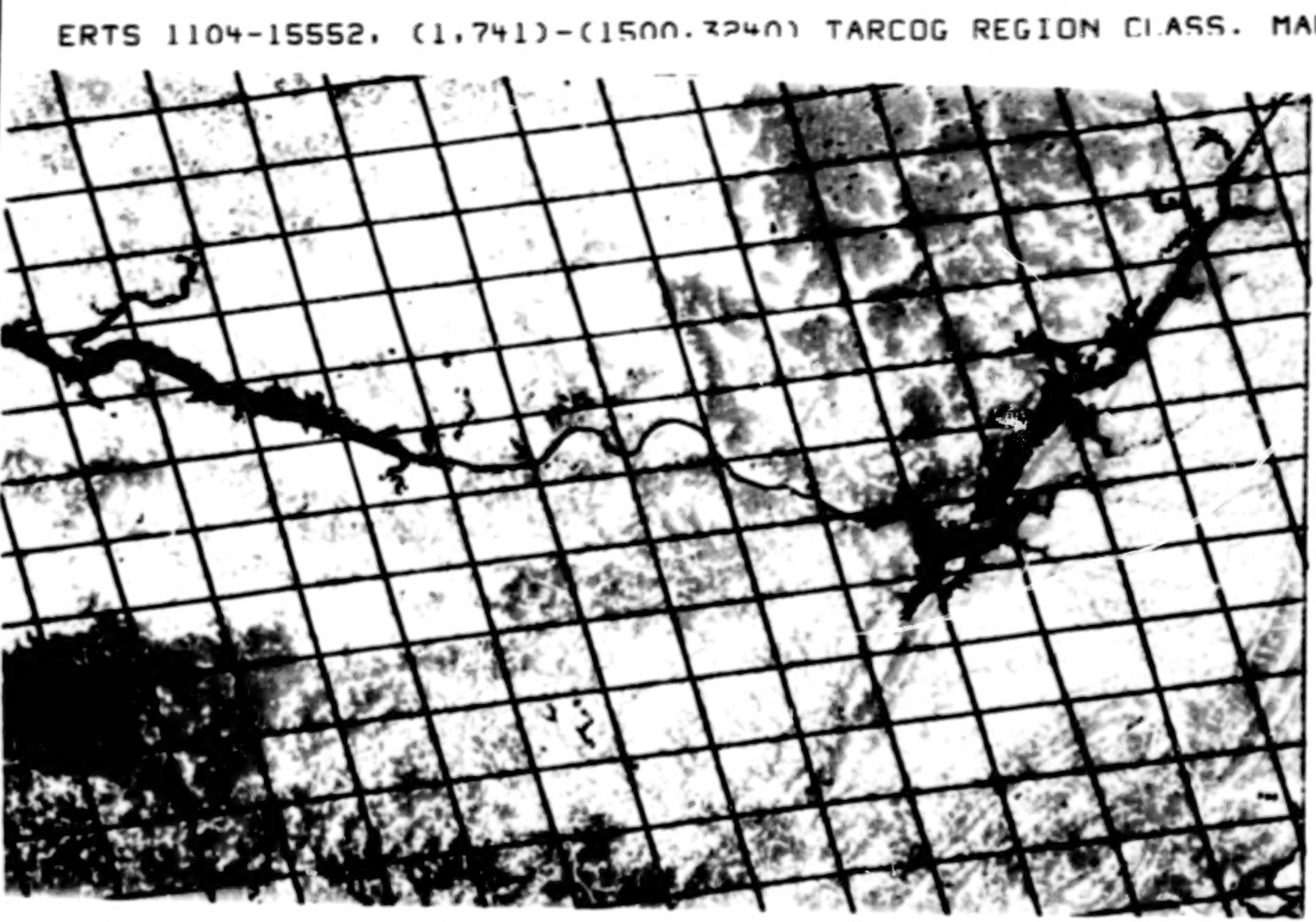


Figure 44. UTM Grid Superimposed on Uncorrected TARCOG Scene.

$$\begin{bmatrix} -2.363 & -12.271 \\ 16.784 & -4.231 \end{bmatrix}.$$

The percentage differences of each element in the theoretical matrix are

$$\begin{bmatrix} 0.80\% & 1.49\% \\ -1.54\% & 1.27\% \end{bmatrix}$$

due to the approximations in the theoretical matrix.

4-4. GEOMETRIC CORRECTION

The transformation determined by least squares minimization was applied to the Huntsville area data and the classification maps. The red spectral band image of the Huntsville area after geometric correction is shown in Figure 45. A segment of data sized 80 lines by 100 samples containing the Huntsville Madison County Jetport is shown in Figure 46. Cubic convolution was used, and the scale was chosen to obtain magnification of the image. The axis labels are kilometers in the UTM system. The geometrically corrected four class map with UTM grid superimposed is shown in Figure 47. A land use map of urban and built-up areas as shown in Figure 48 reveals the locations of cities and major roads and airports. The results may also be tabulated in terms of UTM cells of various sizes, as is illustrated in Figure 49.

Figure 45. Red Band Coverage of Huntsville Region, Geometrically Corrected.

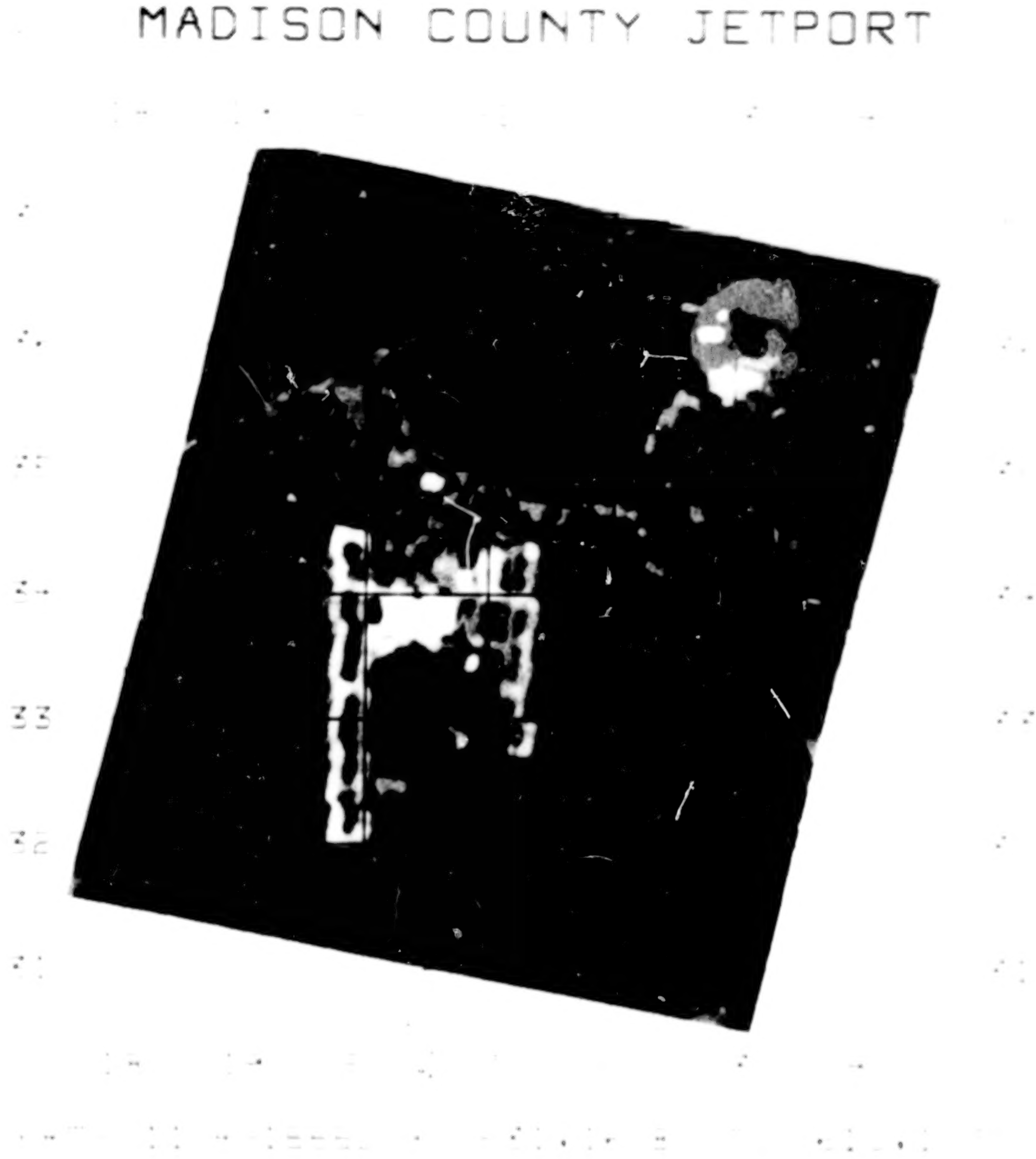


Figure 46. Coverage of Huntsville-Madison County Jetport, Geometrically Corrected and Magnified by Cubic Interpolation.

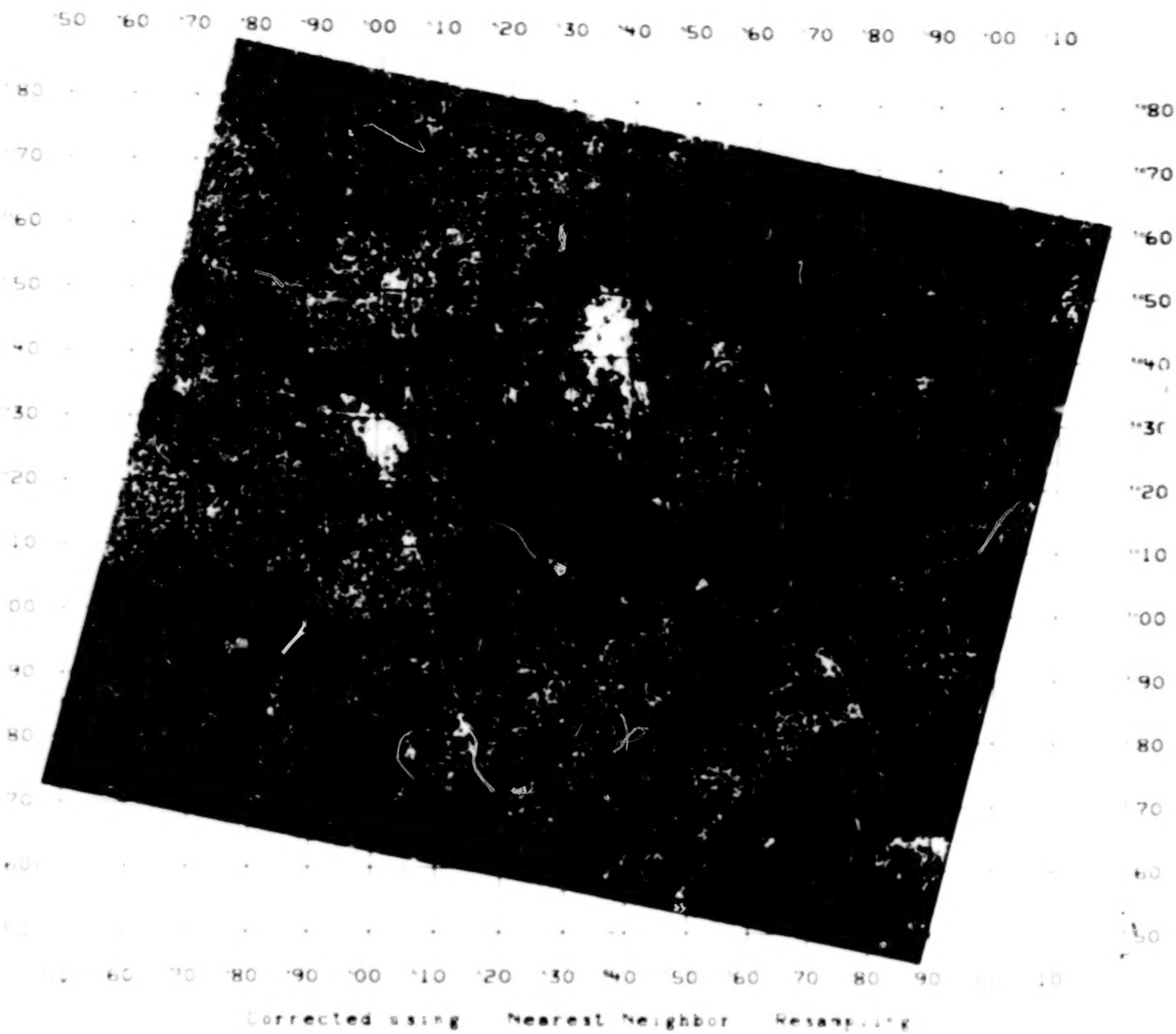


Figure 47. Four Class Map of TARCOG Region Geometrically Corrected with UTM Grid Superimposed.

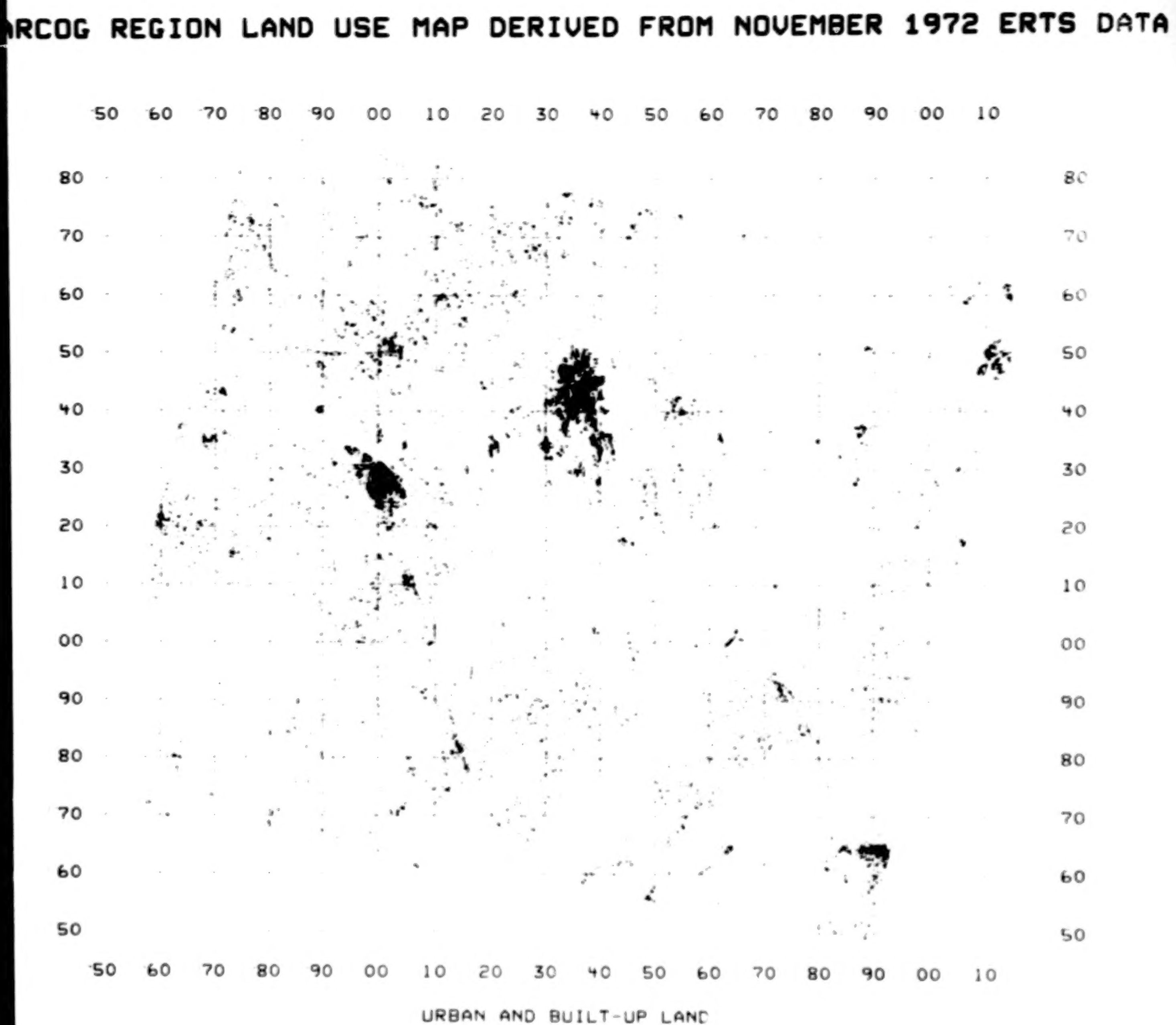


Figure 48. Urban Land use Areas in TARCOG.

LAND USE CLASSIFICATION SUMMARY

CONTENTS OF 10 KM BY 10 KM UTM CELL WITH SOUTHWEST CORNER (560., 3760.)

URBAN 4.114 PERCENT

AGRICULTURE 23.315 PERCENT

FOREST 71.917 PERCENT

WATER 0.653 PERCENT

STATISTICS BASED ON ANALYSIS OF 7656 SAMPLES

BREAKDOWN BY 1 KM SQUARE CELLS
PERCENTAGE OCCUPANCY

URBAN	2.8	2.8	0.0	1.4	11.1	0.0	3.1	1.4	4.2	1.4
AGRICULTURE	1.4	11.1	3.1	23.6	59.7	16.7	25.0	20.8	38.9	15.3
FOREST	95.8	86.1	96.9	75.0	29.2	83.3	71.9	76.4	56.9	83.3
WATER	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.4	0.0	0.0
URBAN	1.2	1.2	8.3	8.6	1.2	0.0	0.0	0.0	0.0	0.0
AGRICULTURE	16.0	4.9	22.2	39.5	32.1	42.0	38.9	56.8	6.2	16.0
FOREST	82.7	93.8	68.1	49.4	66.7	58.0	61.1	43.2	93.8	84.0
WATER	0.0	0.0	1.4	2.5	0.0	0.0	0.0	0.0	0.0	0.0
URBAN	0.0	14.8	1.4	1.2	2.5	2.5	0.0	0.0	1.2	3.7
AGRICULTURE	46.9	54.3	45.8	17.3	39.5	16.0	15.3	17.3	19.8	30.9
FOREST	53.1	28.4	38.9	79.0	58.0	81.5	84.7	82.7	79.0	65.4
WATER	0.0	2.5	13.9	2.5	0.0	0.0	0.0	0.0	0.0	0.0
URBAN	2.5	2.5	8.3	1.2	0.0	1.2	2.8	0.0	3.7	4.9
AGRICULTURE	43.2	39.5	48.6	27.2	12.3	12.3	9.7	7.4	45.7	43.2
FOREST	54.3	56.8	43.1	71.6	87.7	86.4	87.5	91.4	50.6	51.9
WATER	0.0	1.2	0.0	0.0	0.0	0.0	0.0	1.2	0.0	0.0
URBAN	0.0	0.0	13.9	2.5	4.9	6.2	0.0	0.0	6.2	3.7
AGRICULTURE	16.0	39.0	45.8	19.8	14.8	12.3	6.9	24.7	45.7	46.9
FOREST	84.0	61.0	40.3	77.8	80.2	81.5	93.1	69.1	48.1	49.4
WATER	0.0	0.0	0.0	0.0	0.0	0.0	0.0	6.2	0.0	0.0
URBAN	0.0	6.9	20.3	65.3	38.9	0.0	1.6	8.3	0.0	5.6
AGRICULTURE	31.9	29.2	37.5	16.7	26.4	25.0	34.4	44.4	20.8	20.8
FOREST	68.1	63.9	42.2	18.1	34.7	75.0	64.1	47.2	79.2	73.6
WATER	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
URBAN	0.0	4.9	19.4	28.4	1.2	0.0	0.0	2.5	0.0	1.2
AGRICULTURE	19.8	17.3	43.1	18.5	30.9	30.9	6.9	19.8	7.4	13.6
FOREST	80.2	77.8	37.5	53.1	67.9	69.1	93.1	77.8	90.1	85.2
WATER	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.5	0.0
URBAN	2.5	8.6	12.5	0.0	0.0	0.0	0.0	0.0	0.0	3.7
AGRICULTURE	11.1	39.5	25.0	9.9	22.2	11.1	1.4	3.7	11.1	17.3
FOREST	81.5	51.9	62.5	90.1	77.8	88.9	98.6	96.3	88.9	79.0
WATER	4.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
URBAN	8.6	2.5	5.6	4.9	3.7	1.2	0.0	0.0	2.5	8.6
AGRICULTURE	25.9	4.9	31.9	28.4	18.5	6.2	9.7	3.7	28.4	35.8
FOREST	65.4	92.6	62.5	66.7	77.8	92.6	90.3	92.6	69.1	55.6
WATER	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.7	0.0	0.0
URBAN	0.0	0.0	0.0	1.4	0.0	0.0	0.0	2.8	4.2	1.4
AGRICULTURE	2.8	31.9	21.9	13.9	2.8	19.4	9.4	8.3	16.7	5.6
FOREST	97.2	68.1	78.1	84.7	97.2	80.6	73.4	81.9	79.2	93.1
WATER	0.0	0.0	0.0	0.0	0.0	0.0	17.2	6.9	0.0	0.0

Figure 49. Classification summary of a 10 km. by 10 km. UTM cell.

4-4-1 EFFECT OF RESAMPLING ON CLASSIFICATION

The uncorrected ERTS MSS data, while good for visual identification of gross ground features, require geometric corrections for comparison with standard maps and images from other sensors. Exact resampling can be performed using sinc functions (under the assumption of band-limitedness of the data), but practical considerations require approximate interpolation to produce the radiometric values in the geometrically corrected images. Two approaches can be used to study the radiometric fidelity of such images. The errors relative to $\sin(x)/x$ function interpolation are studied or the effects of interpolation on the performance of classifier are experimentally evaluated. It is seen that the overall class occupancy statistics change only slightly, but the point-by-point differences between the classifications of corrected and uncorrected data are noticeable.

In order to study the effect of resampling for geometric correction on classification accuracy, it is necessary to compare the classifications before and after geometric correction with the ground truth. With supervised classification, however, the actual classifications depend on the choice of training samples. Therefore, to have a uniform basis for comparing the classification performance, it is desirable to use the same set of training samples for the "before" and "after" classifications.

Several types of before-and-after comparisons can be made. The ground truth is difficult to gather and convert into machine-readable format for areas large enough to be statistically significant. Therefore, the classification map of the uncorrected data based on training samples chosen as accurately as possible is chosen as a standard. When this classification map is geometrically corrected using the nearest neighbor rule for resampling, the resulting map can be used for point-by-point comparison with the classification of geometrically corrected data. This map can be compared with the classification maps obtained when geometric correction is made using linear or cubic interpolation and the training is performed using samples from the uncorrected image or the corresponding locations in the corrected image. While such comparisons do not show which type of classification is the most accurate, they do indicate whether the effect of geometric correction is significant.

One such study was made on a 500 x 500 pixel four-band Landsat image of a region containing Huntsville, Alabama. A map showing four land use classes (urban, agriculture, forest, water) was obtained using a sequential linear classifier whose discriminant hyperplanes were obtained by training on the raw data samples from each of these classes. The classification map was geometrically corrected for the heading angle and earth rotation effects using nearest neighbor values after resampling. Also, the four bands were individually corrected using the same correction transformation, but using linear and cubic interpolation rules. Four classification maps were produced, two with the original training

samples and two more with training samples taken from the geometrically corrected images. The following abbreviations will be introduced to facilitate reference to the experimental results.

C = Classification map of the uncorrected data.

NN = Result of geometrically correcting C using nearest neighbor values.

L/U = Classification map of the geometrically corrected data using linear interpolation/training samples being from the uncorrected image.

C/U = Classification map of the geometrically corrected data using cubic interpolation/training samples being from the uncorrected image.

L/L = Same as L/U, except that the training samples are from the corrected image.

C/C = Same as C/U, except that the training samples are from the corrected image.

The four classes are denoted by

1 = Urban

2 = Agriculture

3 = Forest

4 = Water

When images are geometrically corrected, in general, they become non-rectangular with edges not parallel and perpendicular to the scan lines. For convenience, they are stored in a circumscribing rectangle. Therefore, there are several points in the corrected images files which do not belong in the images. These points are indicated by the class number 0.

Tables 10 through 15 indicate the number of occurences of each of the classes 0 through 4 in the various classification maps. It can be seen that there is no significant change in the percentage occupancy of each of the classes (1 through 4).

The point-by-point differences between the classification maps can be summarized in various ways. Let D(X,Y) denote the 5x5 matrix whose ij^{th} element consists of the number of occurrences of the i^{th} class in image X and j^{th} class in image Y at the same location. Then, the matrices of the NN corrected classification vs. classifications of corrected data are shown in Tables 16-19.

Table 10. Class Occupancy in C

Class	Number of Samples	Percentage
Urban	28475	11.39
Agriculture	111196	44.48
Forest	104978	41.99
Water	5351	2.14

Table 11. Class Occupancy in NN

Class	Number of Samples	Percentage
Urban	40559	11.43
Agriculture	157644	44.42
Forest	149089	42.01
Water	7609	2.14

Table 12. Class Occupancy in L/U

Class	Number of Samples	Percentage
Urban	37781	10.65
Agriculture	165926	46.75
Forest	144186	40.63
Water	7008	1.97

Table 13. Class Occupancy in L/L

Class	Number of Samples	Percentage
Urban	35825	10.09
Agriculture	166760	46.99
Forest	144043	40.59
Water	8273	2.33

Table 14. Class Occupancy in C/U

Class	Number of Samples	Percentage
Urban	40200	11.33
Agriculture	158971	44.79
Forest	148011	41.70
Water	7719	2.17

Table 15. Class Occupancy of C/C

Class	Number of Samples	Percentage
Urban	38383	10.82
Agriculture	160465	45.21
Forest	147489	41.56
Water	8564	2.41

Table 16. Matrix D(NN, L/U)

Class in NN	Class in L/U				
	0	1	2	3	4
0	164243	0	0	0	0
1	0	33656	6808	14	81
2	0	3837	146598	7133	76
3	0	52	12357	136420	260
4	0	236	163	619	6591

Table 17. Matrix D(NN, L/C)

Class in NN	Class in L/C				
	0	1	2	3	4
0	164243	0	0	0	0
1	0	32580	7764	16	199
2	0	3081	146497	7889	177
3	0	35	12366	135757	931
4	0	129	133	381	6966

Table 18. Matrix D(NN, C/U)

Class in NN	Class in C/U				
	0	1	2	3	4
0	164243	0	0	0	0
1	0	34463	5884	40	172
2	0	5489	141718	10295	142
3	0	90	11234	137197	568
4	0	158	135	479	6837

Table 19. Matrix D(NN, C/C)

Class in NN	Class in C/C				
	0	1	2	3	4
0	164243	0	0	0	0
1	0	33647	6654	37	221
2	0	4552	142420	10477	195
3	0	75	11265	136650	1099
4	0	109	126	325	7019

Note that in these matrices the off-diagonal elements are generally much smaller than the corresponding diagonal elements. Also, the ij^{th} and ji^{th} elements are of the same order for all i and j. This accounts for the smallness in the percentage differences in class occupancies between the classification maps.

The differences between NN and L/U or C/U are caused solely by the interpolation process since the training samples used are identical and hence the discriminant hyperplanes are also identical. The dependence of the classifications in L/U or C/U on interpolation can be illustrated as follows. The feature vector at any point A in the geometrically corrected image is obtained by interpolation from 4 (or 16) feature vectors in the uncorrected image at the points on a 2x2 (or 4x4) array surrounding the point corresponding to A. The feature vectors participating in interpolation may not all be in one class. The classes that do enter into interpolation can be found by applying the geometric correction to the classification map and, instead of using any type of interpolation, generating a unique number indicative of the class combinations in the 2x2 (or 4x4) array. A matrix of the type shown in Tables 16 through 19 can be obtained for each subset of points in the image having a given class combination. Such matrices are shown for all class combinations possible showing differences between NN and L/U in Table 20. In this table the class combination (n_1 n_2 n_3 n_4) indicates that n_i feature vectors from class i entered into interpolation. The "nearest neighbor" is the value in NN. The table, then, consists of the number of points with interpolation class combination (n_1 n_2 n_3 n_4) and NN value i that got classified as j in L/U. Thus, it can be seen that there were a total of 30,391 points at which the class combination (0 2 2 0) occurred (i.e., interpolation was between two samples each of agriculture and forest) and 2483 of them were classified into the forest class, even though the nearest neighbors were in the agriculture class. Some general observations can be made from this table.

(i) When only one class enters into interpolation, all but a negligible percentage of points in L/U fall into that class.

(ii) When two classes enter into interpolation, a significant portion of points in L/U might belong to classes other than the two classes involved (e.g., (2 0 2 0), (0 2 0 2) and (0 0 2 2)).

(iii) When more than one class enters into interpolation, the nearest neighbors tend to dominate the classifications in L/U.

These empirical conclusions are easily justified from theoretical considerations. For, a feature vector q in the geometrically corrected image using linear interpolation is obtained by

$$q = (\alpha p_1 + (1 - \alpha) p_2)\beta + (\alpha p_3 + (1 - \alpha) p_4) (1 - \beta)$$

Table 20. Class Occupancy vs. Combinations for Interpolation in L/U

CLASS COMBINATION	NEAREST NEIGHBOR	NUMBER OF OCCURENCES* CLASS 0	CLASS 1	CLASS 2	CLASS 3	CLASS 4
0 0 0 0	0	164243	0	0	0	0
	1	0	0	0	0	0
	2	0	0	0	0	0
	3	0	0	0	0	0
	4	0	0	0	0	0
4 0 0 0	0	0	0	0	0	0
	1	0	17157	37	0	0
	2	0	0	0	0	0
	3	0	0	0	0	0
	4	0	0	0	0	0
3 1 0 0	0	0	0	0	0	0
	1	0	7437	417	0	0
	2	0	1471	1188	0	0
	3	0	0	0	0	0
	4	0	0	0	0	0
2 2 0 0	0	0	0	0	0	0
	1	0	5586	1791	0	0
	2	0	1440	5106	0	0
	3	0	0	9	0	0
	4	0	0	0	0	0
1 3 0 0	0	0	0	0	0	0
	1	0	1867	3057	0	0
	2	0	611	14204	1	0
	3	0	0	0	0	0
	4	0	0	0	0	0
0 4 0 0	0	0	0	0	0	0
	1	0	0	0	0	0
	2	0	68	84018	89	0
	3	0	0	0	0	0
	4	0	0	0	0	0
3 0 1 0	0	0	0	0	0	0
	1	0	117	16	0	0
	2	0	0	0	0	0
	3	0	11	33	3	1
	4	0	0	0	0	0
2 1 1 0	0	0	0	0	0	0
	1	0	474	262	0	0
	2	0	37	264	4	0
	3	0	9	285	60	0
	4	0	0	0	0	0
1 2 1 0	0	0	0	0	0	0
	1	0	260	601	0	0
	2	0	56	1581	25	0
	3	0	0	659	183	0
	4	0	0	0	0	0
0 3 1 0	0	0	0	0	0	0
	1	0	0	0	0	0
	2	0	5	22433	663	0
	3	0	0	5053	2761	0
	4	0	0	0	0	0

THE NUMBER OF OCCURENCES REFER TO THE OCCURENCE OF CLASSES 0, 1, 2, 3, 4 IN THE CLASSIFICATION MAP OF THE GEOMETRICALLY CORRECTED DATA AT LOCATIONS HAVING THE CORRESPONDING CLASS COMBINATIONS AND NEAREST NEIGHBORS AS DETERMINED USING THE CLASSIFICATION MAP OF THE UNCORRECTED DATA

Table 20. (Continued)

CLASS COMBINATION	NEAREST NEIGHBOR	NUMBER OF OCCURENCES CLASS 0	CLASS 1	CLASS 2	CLASS 3	CLASS 4
2 0 2 0	0	0	0	0	0	0
	1	0	171	130	1	0
	2	0	0	0	0	0
	3	0	13	205	93	0
	4	0	0	0	0	0
1 1 2 0	0	0	0	0	0	0
	1	0	125	360	3	0
	2	0	21	475	25	0
	3	0	1	482	521	0
	4	0	0	0	0	0
0 2 2 0	0	0	0	0	0	0
	1	0	0	0	0	0
	2	0	0	12917	2483	0
	3	0	0	3961	11549	1
	4	0	0	0	0	0
1 0 3 0	0	0	0	0	0	0
	1	0	26	69	7	0
	2	0	0	0	0	0
	3	0	0	85	250	0
	4	0	0	0	0	0
0 1 3 0	0	0	0	0	0	0
	1	0	0	0	0	0
	2	0	0	3052	3530	0
	3	0	0	1274	18438	1
	4	0	0	0	0	0
0 0 4 0	0	0	0	0	0	0
	1	0	0	0	0	0
	2	0	0	0	0	0
	3	0	0	88	100240	11
	4	0	0	0	0	0
3 0 0 1	0	0	0	0	0	0
	1	0	149	1	0	5
	2	0	0	0	0	0
	3	0	0	0	0	0
	4	0	26	0	0	11
2 1 0 1	0	0	0	0	0	0
	1	0	70	3	0	2
	2	0	13	17	1	1
	3	0	0	0	0	0
	4	0	22	3	0	18
1 2 0 1	0	0	0	0	0	0
	1	0	25	9	0	5
	2	0	18	79	1	0
	3	0	0	0	0	0
	4	0	20	7	0	13
0 3 0 1	0	0	0	0	0	0
	1	0	0	0	0	0
	2	0	13	142	1	0
	3	0	0	0	0	0
	4	0	9	25	6	14

THE NUMBER OF OCCURENCES REFER TO THE OCCURENCE OF CLASSES 0, 1, 2, 3, 4 IN THE CLASSIFICATION MAP OF THE GEOMETRICALLY CORRECTED DATA AT LOCATIONS HAVING THE CORRESPONDING CLASS COMBINATIONS AND NEAREST NEIGHBORS AS DETERMINED USING THE CLASSIFI- CATION MAP OF THE UNCORRECTED DATA

Table 20. (Continued)

CLASS COMBINATION	NEAREST NEIGHBOR	NUMBER OF OCCURENCES CLASS 0	CLASS 1	CLASS 2	CLASS 3	CLASS 4
2 0 1 1	0	0	6	0	0	0
	1	0	20	4	1	2
	2	0	6	0	0	0
	3	0	3	7	3	0
	4	0	7	1	2	3
1 1 1 1	0	0	0	0	0	0
	1	0	17	18	0	4
	2	0	11	28	4	1
	3	0	1	19	17	2
	4	0	14	8	1	9
0 2 1 1	0	0	0	0	0	0
	1	0	0	0	0	0
	2	0	2	135	23	0
	3	0	0	29	47	2
	4	0	5	30	34	17
1 0 2 1	0	0	0	0	0	0
	1	0	11	15	1	1
	2	0	0	0	0	0
	3	0	2	24	43	2
	4	0	6	5	5	13
0 1 2 1	0	0	0	0	0	0
	1	0	0	0	0	0
	2	0	1	51	43	3
	3	0	0	25	187	8
	4	0	5	19	64	35
0 0 3 1	0	0	0	0	0	0
	1	0	0	0	0	0
	2	0	0	0	0	0
	3	0	0	10	1339	23
	4	0	7	10	295	146
2 0 0 2	0	0	0	0	0	0
	1	0	80	1	0	14
	2	0	0	0	0	0
	3	0	0	0	0	0
	4	0	22	0	0	84
1 1 0 2	0	0	0	0	0	0
	1	0	[illegible]	12	0	6
	2	0	26	14	0	8
	3	0	0	0	0	0
	4	0	16	2	0	79
0 2 0 2	0	0	0	0	0	0
	1	0	0	0	0	0
	2	0	30	98	12	14
	3	0	0	0	0	0
	4	0	19	10	13	101
1 0 1 2	0	0	0	0		0
	1	0	14	5		2
	2	0	0	0	0	0
	3	0	1	16	8	4
	4	0	6	2	2	41

"THE NUMBER OF OCCURENCES" REFER TO THE OCCURENCE OF CLASSES 0, 1, 2, 3, 4 IN THE CLASSIFICATION MAP OF THE GEOMETRICALLY CORRECTED DATA AT LOCATIONS HAVING THE CORRESPONDING CLASS COMBINATIONS AND NEAREST NEIGHBORS AS DETERMINED USING THE CLASSIFICATION MAP OF THE UNCORRECTED DATA

Table 20. (Continued)

CLASS COMBINATION	NEAREST NEIGHBOR	NUMBER OF OCCURENCES CLASS 0	CLASS 1	CLASS 2	CLASS 3	CLASS 4
0 1 1 2	0	0	0	0	0	0
	1	0	0	0	0	0
	2	0	6	60	25	13
	3	0	3	29	59	11
	4	0	7	15	22	146
0 0 2 2	0	0	0	0	0	0
	1	0	0	0	0	0
	2	0	0	0	0	0
	3	0	6	45	545	92
	4	0	21	16	137	509
1 0 0 3	0	0	0	0	0	0
	1	0	22	0	1	40
	2	0	0	0	0	0
	3	2	0	0	0	0
	4	0	11	1	1	179
0 1 0 3	0	[illegible]	0	0	0	0
	1	0	0	0	0	0
	2	0	14	16	5	36
	3	0	0	0	0	0
	4	0	6	4	2	195
0 0 1 3	0	0	0	0	0	0
	1	0	0	0	0	0
	2	0	0	0	0	0
	3	0	2	8	94	102
	4	0	5	5	33	560
0 0 0 4	0	0	0	0	0	0
	1	0	0	0	0	0
	2	0	0	0	0	0
	3	0	0	0	0	0
	4	0	0	0	2	4418

where p_1, p_2, p_3, p_4 are feature vectors in the uncorrected image and α, β are constants between 0 and 1. The vectors p_1, p_2, p_3, p_4 are classified using the rule

"Assign p to class k iff $(d'_\lambda p + d_{0\lambda}) < 0$ for $\lambda < k$ and $(d'_k p + d_{0k}) > 0$"

where d_λ is a vector and $d_{0\lambda}$ is a scalar defining discriminants for each λ. Now if p_1 and p_2 are assigned to classes i and j, it is easy to see that $\alpha p_1 + (1 - \alpha)p_2$ cannot be assigned to any class number less than Min(i, j). Also, if p_1 and p_2 are assigned to a class i, it is found that the discriminant conditions for class i are satisfied by $\alpha p_1 + (1 - \alpha) p_2$ also.

In producing C, NN and L/U, the order of testing discriminant functions was water, forest, agriculture and urban. Remembering this and examining Table 20, the above theoretical conclusions are confirmed (except for a few anomalies caused, possibly, by round-off errors).

In conclusion, the overall statistics of class occupancy are only negligibly affected by geometric correction. But the effects on a pixel-by-pixel level are noticeable and the differences between the classifications of corrected and uncorrected data tend to compensate such that the overall class occupancies stay approximately the same. Some pecularities may be introduced by interpolation such as obtaining an urban pixel from samples which were classified as forest and water in the uncorrected image. The correct classifications in those cases can only be found by comparison with the ground truth for those locations. Almost all the differences occur at locations where more than one class is involved in interpolation. It is precisely at these points that the raw data from the spacecraft would consist of mixtures of reflectances from different classes. Therefore, it might well be that if the radiometric values at the resampled coordinate locations are estimated accurately (as with a sinc function or cubic convolution) then the resulting classifications would be more accurate than those obtained in NN. Further tests along these lines (other than ground truth surveys) could be made using mixture proportion estimation methods [18, 19].

4-5. SUPERPOSITION OF BOUNDARIES

The final step in the generation of the TARCOG land use map was the superposition of the county boundaries, as shown in Figure 25, after geometric correction to UTM coordinates. The complete seven class land use map is shown in Figure 50.

Using these county boundaries, it was determined, for example, that the forest area in Madison County covers 590 square kilometers or 146 thousand acres.

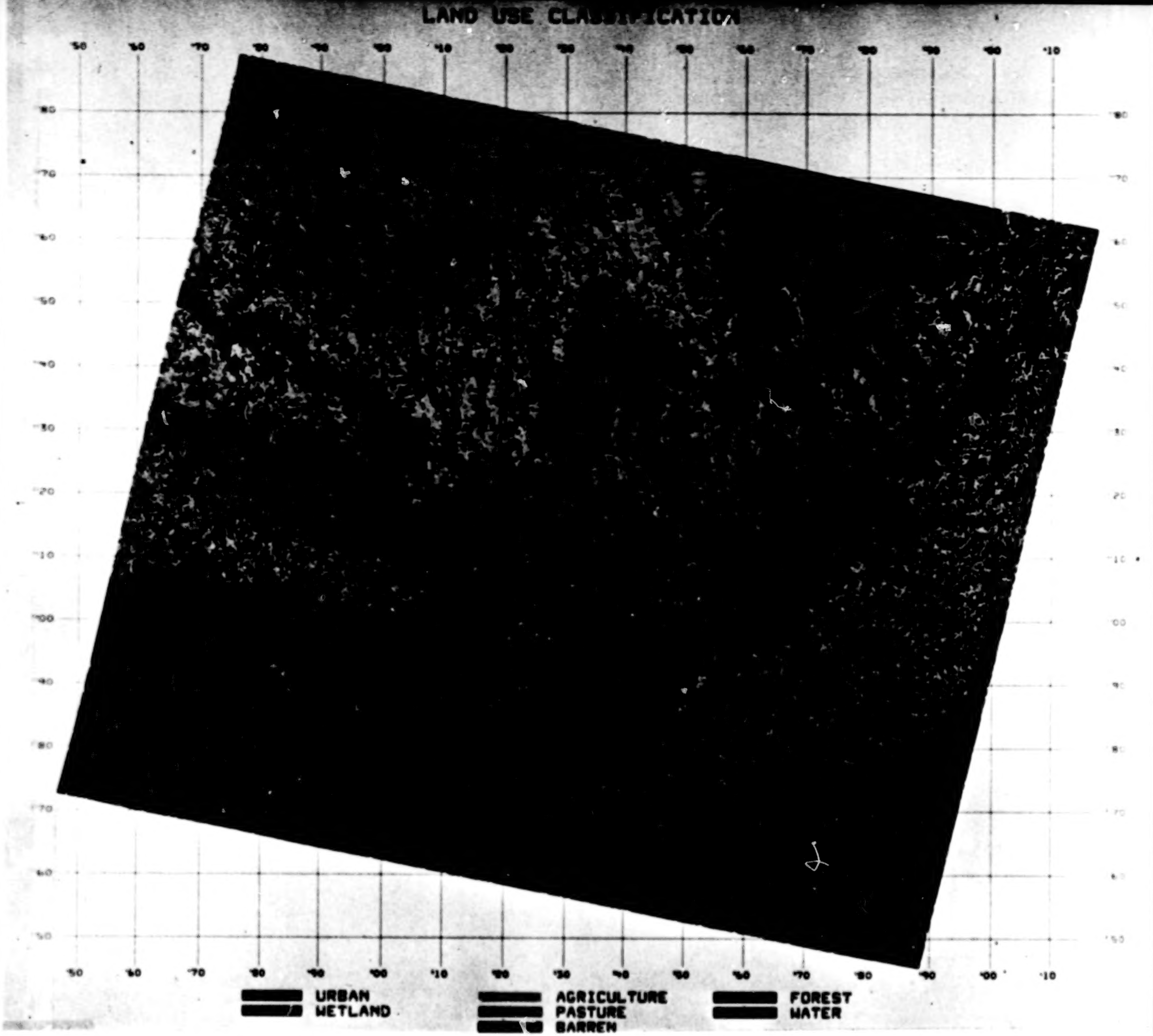


Figure 50. Seven class TARCOG land use map with UTM grids and county boundaries superimposed.

V. COMPUTER PROGRAM DOCUMENTATION

5-1. PRELIMINARY DATA HANDLING

1. NAME

ERTXTM

2. PURPOSE

Extraction and reformatting of a desired rectangular region from four files (corresponding to the four strips) of Landsat data on CCT's. The desired region is specified in terms of latitude and longitude or pixel coordinates.

3. CALLING SEQUENCE

This is a main program. It is currently on a partitioned data set as an executable module.

4. INPUT-OUTPUT

4.1 Input

The following input parameters should be supplied in data cards according to the formats and read statements indicated below.

```
      READ 500,JFLG,KFLG
      READ 500,NBDS,(IBDS(I),I=1,NBDS)
      READ 820,FVECT,BANDS,HIST
      IF(JFLG.EQ.0)READ 500,IRI,IRF,ICI,ICF
      IF(JFLG.GT.0)READ 600,RLATI,RLATF,RLNGI,RLNGF
500   FORMAT(12I6)
600   FORMAT(6F12.3)
820   FORMAT(3L6)
```

where

JFLG,KFLG are "task-indicator" flags.

JFLG=0 indicates that the region is specified by pixel coordinates and should be extracted.

JFLG=1 indicates that the region is specified by geographic coordinates, the corresponding pixel coordinates are to be found and printed, but the region should not be extracted.

JFLG=2 indicates that the region is specified by geographic coordinates, the corresponding pixel coordinates are to be found and printed, and the region should be extracted.

KFLG=0(1) indicates that while extracting, the "synthetic" pixels (which are extra pixels added to adjust the line length) should not (should) be suppressed.

NBDS = number of bands (≤ 4) to be extracted.

IBDS = band numbers (the order specified will dictate the order in which the components in the feature vectors and the individual band files are arranged).

FVECT, BANDS, HIST are logical variables which should be .TRUE. to indicate that a feature vector file, individual band files, and histograms of individual bands (respectively) are desired. Histograms are produced only when BANDS.AND.HIST = .TRUE..

IRI, IRF, ICI, ICF are initial and final rows and initial and final columns respectively of the region to be extracted.

RLATI, RLATF, RLNGI, RLNGF are initial and final latitude and initial and final longitude of the region of interest. They should be supplied in degrees (not degrees and minutes, but decimal degrees). Northern latitudes and Eastern longitudes are considered positive.

The data from the Landsat CCT's should be on four separate data sets which can be "OPEN" at the same time. The DDNAMES for these data sets should be TAPEiF01 with i = 1, 2, 3, 4 in order to be compatible with the non-FORTRAN read routine READNL. If the four strips of data are on the same tape, each strip should be copied to a separate tape (or disk file) before using this program.

4.2 Output

The output of this program will consist of printout of the coordinates requested (as illustrated in the attached example) and, depending on FVECT and BANDS, a file of feature vectors and NBDS files of individual band images. These files will be written as unformatted FORTRAN records. The number of records = IRF - IRI + 1 in the feature vector file and IRF - IRI + 2 in the individual band files. The first record in the individual band files consists of NREC, NEL where NREC = IRF - IRI + 1 and NEL = ICF - ICI + 1 where IRI, IRF, ICI, ICF are the values supplied when JFLG=0 or computed from RLATI, RLATF, RLNGI, RLNGF when JFLG=2. When KFLG=1, NEL is the number of pixels in the shortest record after synthetic pixel removal.

The data in the output files are in bytes. Thus, each record of the feature vector file consists of NEL * NBDS bytes, with NBDS bytes per pixel. Each record of the individual band file consists of NEL bytes.

4.3 File Storage

This program requires a direct access work space of 9360 * 3240 bytes (on logical unit 90) to be able to handle the four-band separation for a full (2340 by 3240) frame. This can, however, be reduced when smaller regions are to be extracted. A convenient way to avoid excessive demand on direct access space is to have the DEFINE FILE statement

```
DEFINE FILE 90 (9360,3240,L,IAV)
```

provide for the maximum space, but use a DD card for unit 90 with the SPACE parameter

```
SPACE = (3240, (585,585))
```

5. EXITS

Not applicable.

6. USAGE

The program is in FORTRAN IV and implemented on the IBM 360 using the H compiler. The program, in its executable form, is in the users' library.

7. EXTERNAL INTERFACES

This program calls several routines. The linkage is indicated in the following table.

Calling Program	Programs Called
ERTXTM	STRTMR PET READNL SVSCI TIKBIN SCLREL GEOPIX ERTXT5 ERTXT6 ERTXT7 PRTHST
TIKBIN	IDIGIT
SCLREL	ENDS CROSS
ERTXT5	READNL ERTXT3 ERTXT4 DAWN SAWN
ERTXT6	SVSCI LRHSTG
ERTXT7	READNL LINFIX ERTXT8 SAWN ERTXT9

8. PERFORMANCE SPECIFICATIONS

8.1 Storage

The program is 34844 bytes long, but including external references required and the buffers, this program needs 128K bytes of storage.

8.2 Execution Time

The execution time depends on the size of the image to be extracted. With FVECT = BANDS = HIST = .TRUE., a 1200 x 1200 region can be processed by this program in approximately 4-1/2 minutes.

8.3 I/O Load

None except as specified by Section 4.

8.4 Restrictions

None

9. METHOD

The program reads the ID and annotation records, finds the number of pixels per record (called the adjusted line length) and prints the exposure date and time and the scene ID. If JFLG=0, the pixel coordinates are read from a card. Otherwise, the routines CTRBIN, TIKBIN, and SCLREL are used to determine a transformation matrix A and the skew angle (due to earth's rotation) using the information about center latitude and longitude and the tick mark locations from the annotation record. The geographic coordinates bounding the region of interest are read from a data card and converted pixel coordinate bounds using GEOPIX. If the pixel coordinates exceed the limits (i.e., if IRI < 1, IRF > 2340, ICI < 1 or ICF $>$ adjusted line length), they are changed to the nearest limits. If the synthetic pixel removal flag KFLG is 0, then the region bounded by IRI, IRF, ICI, ICF is extracted using the routine ERTXT5. If KFLG=1, the region is extracted using ERTXT7 which also removes synthetic pixels. When ERTXT5 is used the number of pixels per line of output is ICF - ICI + 1. When ERTXT7 is used there will be fewer pixels per line, the number being computed and printed by the program.

Other than this, there are no external differences between ERTXT5 and ERTXT7. If FVECT = .TRUE., both routines write a feature vector file on unit NTPFV (=13). If BANDS = .TRUE., the individual bands are separated and written on the direct access unit 90. The order in which

the bands (and components of the feature vectors) are written is dictated by IBDS. If BANDS = .TRUE., the routine ERTXT6 is used to read the individual bands from the direct access file and write as separate files on unit 13. If, in addition, HIST = .TRUE., the histograms of the individual bands are found by ERTXT6 and printed by the routine PRTHST.

10. COMMENTS

The details of the subroutines are omitted here. The methods used in most of the routines are quite straightforward and are apparent from the listings. The routine SCLREL uses some simple concepts from elementary geometry. A useful modification to this program (ERTXTM) would be to include specification of geographic coordinates of the vertices of a more general polygon instead of RLATI, RLATF, RLNGI, RLNGF. As it is, the program extracts a rectangle with sides parallel and perpendicular to the scan lines containing the given rectangle and, in some cases, may yield too large a region. The only routine to be changed to include this generalization is GEOPIX.

11. LISTINGS

The listings of the program are attached at the end.

12. TESTS

The program has been tested, used for the extraction of various Landsat data sets with all the options available, and found to operate satisfactorily.

```
C
C      MAIN PROGRAM ....   ERTXTM   ....
C
       DIMENSION RLUC(6,4),RLTLNG(6,4),IFLG(6,4),LTLNG(6,4)
C      JFLG=1:  COMPUTE REGION TO BE EXTRACTED. DO NOT EXTRACT.
C      JFLG=0   NO COMPUTATION OF REGION TO BE EXTRACTED. EXTRACT GIVEN
C           REGION.
C      JFLG=2:   COMPUTE REGION TO BE EXTRACTED AND EXTRACT.
C      WHEN JFLG=0 OR 2, KFLG=0 OR 1 CORRESPOND TO NO OR YES FOR SYNTHE-
C      TIC PIXEL REMOVAL.
C      IF(BANDS) EXTRACT INDIVIDUAL BANDS ON NTAPO.
C      IF(FVECT) EXTRACT FEATURE VECTORS ON NTPFV.
C
C      IF(BANDS.AND.HIST) FIND AND PRINT HISTOGRAMS OF INDIVIDUAL BANDS
C      IN ADDITION TO EXTRACTING THE BANDS.
C      NB=NUMBER OF BANDS TO BE SEPARATED AND/OR FORMED INTO A F. V.
C      IBDS IS THE ARRAY SPECIFYING THE ORDER IN WHICH F. V. COMPONENTS
C      SHD. OCCUR AND/OR THE FILES OF BANDS SHD. BE ARRANGED ON NTAPO.
C
C      NOTE*** THE FOLLOWING DIMENSIONS SHD BE CHECKED W. R. T. THE IMAGE
C      SIZE TO BE EXTRACTED.
       DIMENSION IBDS(4)
       DIMENSION NSAMPS(4)
       LOGICAL*1 IX(13200),IY(13200)
       DIMENSION IH(256,4)
       INTEGER*2 IZ(20)
       DIMENSION A(2,2)
       LOGICAL*1 HIST
       LOGICAL*1 BANDS,FVECT
       COMMON/GEOLIM/RLATI,RLATF,RLNGI,RLNGF
       COMMON/CNTRE/CTLAT,CTLONG,CNTRX,CNTRY
       COMMON/COORD/IRI,IRF,ICI,ICF
       COMMON/DSK90/NRW90,NBYT90
       DATA NTAPO,NTPFV/13,13/
       DEFINE FILE 90(9360,3240,L,IAV90)
       NRW90=9360
       NBYT90=3240
       READ 500,JFLG,KFLG
       WRITE(6,800)JFLG,KFLG
       READ 500,NBDS,(IBDS(I),I=1,NBDS)
       WRITE(6,810)NBDS,(IBDS(I),I=1,NBDS)
       READ 820,FVECT,BANDS,HIST
       WRITE(6,830)FVECT,BANDS,HIST
C
C      FIND NUMBER OF PIXELS IPIXT PER RECORD
C
       CALL STRTMR
       CALL PET(0)
       CALL READNL(IZ,IEND,LRECL,1)
       LRECL2=LRECL/2
       CNTRX=1170.5
```

```
      NPIXS=IPIXT/4
      CNTRY=FLOAT(IPIXT)/2.+.5
C
      CALL READNL(IX,IEND,LRECL,1)
      WRITE(6,400)
      WRITE(6,110)(IX(I),I=1,7)
      WRITE(6,120)(IX(I),I=107,111)
      WRITE(6,130)(IZ(I),I=1,6)
      IF(JFLG.NE.0)GO TO 30
      READ 500,   IRI,IRF,ICI,ICF
      WRITE(6,500)IRI,IRF,ICI,ICF
      GO TO 40
30    CONTINUE
C
C     READ ANNOTATION RECORD
C
C
C     FIND LATITUDE AND LONGITUDE OF FORMAT CENTER
C
      CALL CTRBIN(IX(11),CTLAT,CTLONG)
C
C     FIND TICK MARK COORDINATES
C
      J=1
      K=1
      CALL SVSCI(RLOC,24,0.)
      CALL SVSCI(RLTLNG,24,0.)
      CALL SVSCI(IFLG,24,0)
      CALL SVSCI(LTLNG,24,0)
      DO 20 I=385,615,10
      CALL TIKBIN(IX(I),RLOC(J,K),RLTLNG(J,K),IFLG(J,K),LTLNG(J,K))
      J=J+1
      IF(J.LE.6)GO TO 20
      J=1
      K=K+1
20    CONTINUE
      WRITE(6,100)CNTRX,CNTRY
      WRITE(6,200)CTLAT,CTLONG
      WRITE(6,1004)
      WRITE(6,1005)
      DO 35 I=1,6
35    WRITE(6,300)(RLOC(I,J),LTLNG(I,J),RLTLNG(I,J),J=1,4)
C
C     FIND SATELLITE HEADING (CHARACTERS 70,71,72)
C
      HEADNG=IDIGIT(IX(70),3)
C
C     COMPUTE TRANSFORMATION  MATRIX A AND TANGENT T OF SKEW ANGLE DUE
C     TO EARTH'S ROTATION.
C
      CALL SCLREL(RLOC,RLTLNG,LTLNG,HEADNG,A,T)
C
C     DETERMINE PIXEL COORDINATES OF THE FOUR CORNERS OF THE RECTANGULAR
C     AREA SPECIFIED IN TERMS OF LATITITUDES AND LONGITUDES.
C     HENCE FIND LIMITS OF AREA TO BE EXTRACTED(ICI,ICF,IRI,IRF)
```

```
C
      READ 600,    RLATI,RLATF,RLNGI,RLNGF
      WRITE(6,600)RLATI,RLATF,RLNGI,RLNGF
      CALL GEOPIX(A,T,2340.,FLOAT(IPIXT))
40    CONTINUE
      WRITE(6,700)IRI,IRF,ICI,ICF
C
      IF(1.LE.IRI.AND.IRF.LE.2340.AND.1.LE.ICI.AND.ICF.LE.IPIXT)GO TO 50
      IF(IRI.GT.2340.OR.IRF.LT.1.OR.ICI.GT.IPIXT.OR.ICF.LT.1)STOP
      IF(IRI.LT.1)IRI=1
      IF(IRF.GT.2340)IRF=2340
      IF(ICI.LT.1)ICI=1
      IF(ICF.GT.IPIXT)ICF=IPIXT
      WRITE(6,701)
      WRITE (6,700)IRI,IRF,ICI,ICF
50    CONTINUE
      CALL PET(1)
      IF(JFLG.EQ.1)STOP
C
C     EXTRACT THE AREA FROM APPROPRIATE STRIPS OF ERTS DATA TAPES.
C
      NEL=ICF-ICI+1
      IF(KFLG.EQ.0)CALL ERTXT5(NPIXS,IX,IY,NBDS,NEL,IBDS,NSAMPS,NSTRP,
     .     BANDS,FVECT,NTPEV)
      IF(KFLG.EQ.1)CALL ERTXT7(NPIXS,IX,IY,NBDS,IBDS,NELMIN,BANDS,FVECT,
     .     NTPEV)
      IF(KFLG.EQ.1)NEL=NELMIN
      CALL PET(1)
      IF(NSTRP.EQ.0)STOP
      IF(FVECT)END FILE NTPEV
      IF(.NOT.BANDS)STOP
      CALL ERTXT6(IX,NBDS,NEL,NTAPD,HIST,IH,IRF-IRI+1)
      CALL PET(1)
      IF(.NOT.HIST)STOP
      DO 60 I=1,NBDS
      WRITE(6,1100)IBDS(I)
60    CALL PRTHST(IH(1,I),256)
      CALL PET(1)
      STOP
100   FORMAT(1X,29HPIXEL COORDINATES OF CENTER=(,F7.1,1H,,F7.1,1H)/)
110   FORMAT(' EXPOSURE DATE:'2A1,1X3A1,1X2A1)
120   FORMAT(' TIME:'2A1,'-'2A1,'-'A1,'0')
130   FORMAT(' SCENE/FRAME ID:'6A2)
200   FORMAT(4X,26HLAT. AND LONG. OF CENTER=(,F8.2,1H,,F8.2,1H)//)
300   FORMAT(4(F18.3,1X,I2,F9.2))
400   FORMAT(1H1)
500   FORMAT(12I6)
600   FORMAT(6F12.3)
700   FORMAT(/,5H IRI=,I5,5H IRF=,I5,5H ICI=,I5,5H ICF=,I5)
701   FORMAT( 94H COMPUTED REGION EXCEEDS THAT AVAILABLE ON THE SUPPLIED
     . FRAME.  THE PART EXTRACTED IS GIVEN BY)
800   FORMAT(' JFLG='I2,' 0: EXTRACT SPECIFIED REGION; 1: FIND PIXEL ADD
     .RESSES OF REGION(GIVEN LAT AND LONG.); 2: FIND ADDRESSES AND EXTRA
     .CT'/' KFLG='I2,' 0: DO NOT REMOVE SYNTHETIC PIXELS; 1: REMOVE SYNT
     .HETIC PIXELS.')
```

```
810   FORMAT(' NO.OF BANDS TO BE EXTRACTED='I2,' BANDS REQUIRED:'4I2)
820   FORMAT(12L6)
830   FORMAT(' FEATURE VECTORS REQUIRED?    'L3/
     .       ' INDIVIDUAL BAND FILES REQUIRED?',L6/
     .       ' HISTOGRAMS OF BANDS REQUIRED?    'L3)
1006  FORMAT (57X,18HMSS TICK MARK DATA/57X,18(1H*)////)
1005  FORMAT (15X,8HTOP EDGE,23X,9HLEFT EDGE,21X,10HRIGHT EDGE,21X,11HBO
     .TTOM EDGE/15X,8(1H*),23X,9(1H*),21X,10(1H*),21X,11(1H*)////)
1100  FORMAT(1H1,10X,18HHISTOGRAM FOR BAND,I2)
      END
```

```
      SUBROUTINE PET(I)
      IF(I.NE.0)GO TO 10
      CALL TIMER(ITIME1)
      TTIME=0.
      WRITE(6,200)
200   FORMAT(10X,'BEGINNING TIMING***   TIME NOW IS 0')
      RETURN
10    CALL TIMER(ITIME2)
      TIME=(ITIME2-ITIME1)/100.
      TTIME=TTIME+TIME
      ITIME1=ITIME2
      WRITE(6,100)TIME,TTIME
100   FORMAT(10X'TIME ELAPSED SINCE LAST PRINTING OF TIME='E12.3,
     .     'SEC.,  TOTAL TIME ELAPSED='E12.3,'SEC.')
      RETURN
      END
```

```
READNL START 0   SUBROUTINE READNL(INBUF,MEND,LRECL,NTAPI)
       BC     15,12(15)
          DC  X'7'
       DC     CL7'READNL'
        STM   14,12,12(13)
        BALR  2,0
        USING *,2
         L      3,0(1)           LOAD ADDRESS OF BUFFER
         USING INAREA,3
        L       5,4(1)           LOAD ADDRESS OF MEND
       L      6,8(1)        LOAD ADDRESS OF LRECL
       L        7,12(1) LOAD ADDRESS OF NTAPI
        LR    10,13              1  SET UP
        LA    13,TSAVE           1  LINKAGE FOR
        ST    13,8(0,10)         1  CALLING OTHER
        ST    10,4(0,13)         1  ROUTINES
       L        12,0(0,7)        R12=NTAPI
       SH       12,=H'1'         R12=NTAPI-1
       SLL      12,2             R12=(NTAPI-1)*4
       A        12,=A(OPNTAB)  R12=ADDRESS OF OPNTAB+(NTAPI-1)*4
       L        12,0(0,12)       R12=ADDRESS OF NTAPI'TH INDCB
       OPEN     ((12),(INPUT))
         LA     9,EOFEXIT
         ST     9,EOFADD
       MVC      33(3,12),EOFADD+1
       GET      (12),INBUF
       LH       8,82(12)
       ST     8,0(6)
RETRN    L      13,TSAVE+4
        LM    2,12,28(13)       RESTORE REGISTERS
         L      14,12(13)
        MVI   12(13),X'FF'      SIGNAL COMPLETION OF SUBROUTINE
        BCR   15,14             RETURN
EOFEXIT  L      8,=F'1'
         ST     8,0(5)           SET MEND = 1
       CLOSE    ((12),LEAVE)
         B      RETRN
OPNTAB DC       A(INDCB1,INDCB2,INDCB3,INDCB4,INDCB5,INDCB6)
       DC       A(INDCB7,INDCB8,INDCB9)
       DS       4F
INDCB1 DCB      DDNAME=TAPE1F01,DEVD=TA,DSORG=PS,BUFNO=2,MACRF=(GM),
                EROPT=ACC
INDCB2 DCB      DDNAME=TAPE2F01,DEVD=TA,DSORG=PS,BUFNO=2,MACRF=(GM),
                EROPT=ACC
INDCB3 DCB      DDNAME=TAPE3F01,DEVD=TA,DSORG=PS,BUFNO=2,MACRF=(GM),
                EROPT=ACC
INDCB4 DCB      DDNAME=TAPE4F01,DEVD=TA,DSORG=PS,BUFNO=2,MACRF=(GM),
                EROPT=ACC
INDCB5 DCB      DDNAME=TAPE5F01,DEVD=TA,DSORG=PS,BUFNO=2,MACRF=(GM),
                EROPT=ACC
INDCB6 DCB      DDNAME=TAPE6F01,DEVD=TA,DSORG=PS,BUFNO=2,MACRF=(GM),
                EROPT=ACC
```

```
INDCB7 DCB      DDNAME=TAPE7F01,DEVD=TA,DSORG=PS,BUFNO=2,MACRF=(GM),
                EROPT=ACC
INDCB8 DCB      DDNAME=TAPE8F01,DEVD=TA,DSORG=PS,BUFNO=2,MACRF=(GM),
                EROPT=ACC
INDCB9 DCB      DDNAME=TAPE9F01,DEVD=TA,DSORG=PS,BUFNO=2,MACRF=(GM),
                EROPT=ACC
COUNT     DC    CL38'02030405060708091011121314151617181920'
TSAVE     DS    9D
EOFADD    DS    1F
INAREA  DSECT
INBUF   DS   900F   THIS SIZE CAN BE CHANGED TO DESIRED VALUE
        END
```

```
      FUNCTION IDIGIT(L,N)
      LOGICAL*1 L(N),LW(4)
      EQUIVALENCE (IW,LW(1))
      DATA IW/0/
      IFAC=10**(N-1)
      J=0
      DO 10 I=1,N
      LW(4)=L(I)
      J=J+MOD(IW,16)*IFAC
      IFAC=IFAC/10
   10 CONTINUE
      IDIGIT=J
      RETURN
      END
```

```
      SUBROUTINE CTRBIN(IX,RLAT,RLONG)
C
C     TO CONVERT COORDINATES OF FORMAT CENTER(LAT. AND LONG.) OF ERTS
C     CCT IMAGE TO BINARY.
C
      LOGICAL*1 IX(14)
      LOGICAL*1 IS/'S'/,  IW/'W'/
      RLAT=IDIGIT(IX(2),2)+IDIGIT(IX(5),2)/60.
      RLONG=IDIGIT(IX(9),3)+IDIGIT(IX(13),2)/60.
      IF(IX(1).EQ.IS)RLAT=-RLAT
      IF(IX(8).EQ.IW)RLONG=-RLONG
      RETURN
      END
```

```
      SUBROUTINE TIKBIN(IX,RLOC,RLTLNG,IFLG,LTLNG)
C
C     TO CONVERT LATTITUDE OR LONGITUDE OF TICK MARKS TO FLOATING POI
C     BINARY.
C
      LOGICAL*1 IX(10)
      LOGICAL*1 IN,IS,IE,IW,ITIK1,ITIK2
      INTEGER*2 II
      LOGICAL*1 LW(2)
      EQUIVALENCE (II,LW(1))
      DATA II/0/
       DATA IN,IS,IE,IW,ITIK1,ITIK2/'N','S','E','W','|','='/
      DATA I2TO15/Z8000/
C
C     FIRST FIND LOCATION OF TICK MARK IF ANY.
C
      IFLG=0
      IF(IX(3).EQ.ITIK1.OR.IX(3).EQ.ITIK2)IFLG=4
      IF(IX(10).EQ.ITIK1.OR.IX(10).EQ.ITIK2)IFLG=3
      IF(IFLG.EQ.0)RETURN
C
C     IF(IFLG.EQ.0) THERE IS NO TICK MARK CORR. TO VECTOR IX SUPPLIED
C
      LW(1)=IX(1)
      LW(2)=IX(2)
      RLOC=II/FLOAT(I2TO15)
      IF(IX(IFLG).EQ.IN)LTLNG=1
      IF(IX(IFLG).EQ.IS)LTLNG=-1
      IF(IX(IFLG).EQ.IE)LTLNG=+2
      IF(IX(IFLG).EQ.IW)LTLNG=-2
      RLTLNG=IDIGIT(IX(IFLG+1),3)+IDIGIT(IX(IFLG+5),2)/60.
      IF(LTLNG.LT.0)RLTLNG=-RLTLNG
      LTLNG=IABS(LTLNG)
      RETURN
      END
```

```
      SUBROUTINE SCLREL(RLOC,RLTLNG,LTLNG,HEADNG,A,T)
      REAL ITBL
      DIMENSION RLOC(6,4),RLTLNG(6,4),LTLNG(6,4),A(2,2),ITBL(4)
      COMMON/CNTRE/CLAT,CLNG,CNTRX,CNTRY
C
C     TO FIND THE TRANSFORMATION MATRIX A AND T, THE TANGENT OF THE SKE
C     ANGLE IN THE FILM IMAGE.
C     (LATITUDE )      (X)    (CLAT)
C     (          ) = A (  ) + (      )
C     (LONGITUDE)      (Y)    (CLNG)
C     RLOC,RLTLNG ARE ARRAYS CONTAINING FILM COORDINATES, GEOGRAPHIC
C     COORDINATES AND LATITUDE/LONGITUDE INDICATORS CORR.TO TICK MARKS.
C     LTLNG(I,J)=1 IF (I,J)'TH TICKMARK IS A LATITUDE AND 2 IF IT IS A
C     LONGITUDE.  J=1,2,3,4 FOR TOP, LEFT, RIGHT, AND BOTTOM EDGES OF TH
C     IMAGE ON FILM.
C
      CALL ENDS(LTLNG(1,1),6,2,IT1,IT2)
      CALL ENDS(LTLNG(1,4),6,2,IB1,IB2)
      DTIK=RLOC(IT2,1)-RLOC(IT1,1)+RLOC(IB2,4)-RLOC(IB1,4)
      DANG=RLTLNG(IT2,1)-RLTLNG(IT1,1)+RLTLNG(IB2,4)-RLTLNG(IB1,4)
      A(1,1)=DANG/DTIK
C
      CALL ENDS(LTLNG(1,2),6,1,IL1,IL2)
      CALL ENDS(LTLNG(1,3),6,1,IR1,IR2)
      DANG=RLTLNG(IL2,2)-RLTLNG(IL1,2)+RLTLNG(IR2,3)-RLTLNG(IR1,3)
      DTIK=RLOC(IL2,2)-RLOC(IL1,2)+RLOC(IR2,3)-RLOC(IR1,3)
      A(2,2)=DANG/DTIK
      A(1,2)=0.
      N=0
      ITBL(1)=-1./(.5+5.5/180.)
      ITBL(4)=1./(.5+3.25/180.)
      DO 10 I=1,6
      DO 10 J=1,4,3
      IF(LTLNG(I,J).NE.2)GO TO 10
      N=N+1
      A(1,2)=A(1,2)+(RLTLNG(I,J)-CLNG -A(1,1)*RLOC(I,J))*ITBL(J)
10    CONTINUE
      A(1,2)=A(1,2)/N
C
C     FIND X COORDINATES OF LEFT AND RIGHT EDGES FOR TICK MARKS.
C
      XL=0.
      N=0
      DO 20 I=1,6
      IF(LTLNG(I,2).NE.2)GO TO 20
      XL=XL+(RLTLNG(I,2)-CLNG-A(1,2)*RLOC(I,2))
      N=N+1
20    CONTINUE
      IF(N.NE.0)XL=XL/N/A(1,1)
      IF(N.EQ.0)XL=.55
      XR=0.
```

```
      N=0
      DO 30 I=1,6
      IF(LTLNG(I,3).NE.2)GO TO 30
      XR=XR+(RLTLNG(I,3)-CLNG-A(1,2)*RLOC(I,3))
      N=N+1
30    CONTINUE
      IF(N.NE.0)XR=XR/N/A(1,1)
      IF(N.EQ.0)XR=-.55
      ITBL(2)=1./XL
      ITBL(3)=1./XR
      N=0
      A(2,1)=0.
      DO 40 I=1,6
      DO 40 J=2,3
      IF(LTLNG(I,J).NE.1)GO TO 40
      N=N+1
      A(2,1)=A(2,1)+(RLTLNG(I,J)-CLAT-A(2,2)*RLOC(I,J))*ITBL(J)
40    CONTINUE
      A(2,1)=A(2,1)/N
      DEGRAD=ATAN(1.)/45.
      H=(HEADNG-180.)*DEGRAD
      T=18./251.*COS(CLAT*DEGRAD)
      T=T*COS(H)/(1.-T*SIN(H))
      SKEW=ATAN(T)*180./3.1415962
      PRINT 100,((A(I,J),J=1,2),I=1,2),SKEW
100   FORMAT(//' TRANSFORMATION FROM FILM COORDINATES TO GEOGRAPHIC COOR
      DINATES'/2(2E15.6)/' SKEW ANGLE ON FILM='F8.3,' DEGREES')
      PRINT 400
400   FORMAT(//' RESIDUALS AT TICK MARKS WHEN TRANSFORMATION IS USED')
      PRINT 500
500    FORMAT(/' TOP AND BOTTOM EDGES')
      DO 60 J=1,4,3
      Y=1./ITBL(J)
      DO 50 I=1,6
      IF(LTLNG(I,J).NE.2)GO TO 70
C
C     Y AND LONGITUDE ARE GIVEN.
C
      CALL CROSS(A(1,2),A(1,1),A(2,2),A(2,1),Y,RLTLNG(I,J)-CLNG,X,PHI)
      GO TO 75
70    IF(LTLNG(I,J).NE.1)GO TO 50
C
C     Y AND LATITUDE ARE GIVEN.
C
      CALL CROSS(A(2,2),A(2,1),A(1,2),A(1,1),Y,RLTLNG(I,J)-CLAT,X,PHI)
75    DX=RLOC(I,J)-X
      PRINT 600,X,Y,RLTLNG(I,J),PHI,RLOC(I,J),DX
600   FORMAT(6F15.5)
50    CONTINUE
60    CONTINUE
      PRINT 700
700   FORMAT(/' LEFT AND RIGHT EDGES')
      DO 80 J=2,3
      X=1./ITBL(J)
      DO 80 I=1,6
```

```
      IF(LTLNG(I,J).NE.1)GO TO 90
C
C     X AND LATITUDE ARE GIVEN.
C
      CALL CROSS(A(2,1),A(2,2),A(1,1),A(1,2),X,RLTLNG(I,J)-CLAT,Y,PHI)
      GO TO 95
90    IF(LTLNG(I,J).NE.2)GO TO 80
C
C     X AND LONGITUDE ARE GIVEN
C
      CALL CROSS(A(1,1),A(1,2),A(2,1),A(2,2),X,RLTLNG(I,J)-CLNG,Y,PHI)
95    DY=RLOC(I,J)-Y
      PRINT 600,X,Y,RLTLNG(I,J),PHI,RLOC(I,J),DY
80    CONTINUE
      RETURN
200   FORMAT(20X4I10)
300   FORMAT(10X3F15.5)
      END
```

```
      SUBROUTINE ENDS(IX,N,J,I1,I2)
      DIMENSION IX(N)
C
C     FIND I1,I2,THE SMALLEST AND LARGEST INDICES I BETWEEN 1 AND N SUCH THAT
C     IX(I)=J.
C     I1=0 IF NO SUCH I EXISTS.
C
      I1=0
      DO 10 I=1,N
      IF(IX(I).NE.J)GO TO 10
      I1=I
      GO TO 20
10    CONTINUE
      RETURN
20    I2=I1
      I1N=I1+N
      DO 30 I=I1,N
      K=I1N-I
      IF(IX(K).NE.J)GO TO 30
      I2=K
      RETURN
30    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE CROSS(A,B,C,D,X,U,Y,V)
C     SOLVE FOR Y,V GIVEN X,U.
C     U=AX+BY; V=CX+DY
C
      Y=(U-A*X)/B
      V=C*X+D*Y
      RETURN
      END
```

```
      SUBROUTINE GEOPIX(A,T,XUP,YUP)
      COMMON/CNTRE/CLAT,CLNG,CNTRX,CNTRY
      COMMON/GEOLIM/RLATI,RLATF,RLNGI,RLNGF
      COMMON/COORD/IRI,IRF,ICI,ICF
      DIMENSION A(2,2)
      DIMENSION B(2,2)
      DIMENSION CORN(2,4)
C
C     FIND INVERSE OF MATRIX A.
C
      DET=A(1,1)*A(2,2)-A(1,2)*A(2,1)
      B(1,1)=A(2,2)/DET
      B(1,2)=-A(1,2)/DET
      B(2,1)=-A(2,1)/DET
      B(2,2)=A(1,1)/DET
C
C     FIND AND PRINT FILM COORDINATES AND PIXEL INCREMENTS(FROM CENTER)
C     OF THE 4 CORNERS OF THE RECTANGLE TO BE EXTRACTED.
C
      CORMN1=1.E10
      CORMN2=1.E10
      CORMX1=-CORMN1
      CORMX2=-CORMN1
      PRINT 100
      DO 10 I=1,4
      RLAT=RLATI
      IF(I.GT.2)RLAT=RLATF
      RLAT=RLAT-CLAT
      RLNG=RLNGI
      IF(MOD(I,2).EQ.0)RLNG=RLNGF
      RLNG=RLNG-CLNG
      CORN(1,I)=B(1,1)*RLNG+B(1,2)*RLAT
      CORN(2,I)=B(2,1)*RLNG+B(2,2)*RLAT
      CORN1=CORN(2,I)*(XUP-85)
      CORN2=(-CORN(1,I)+CORN(2,I)*T)*(YUP-1)
      CORMN1=AMIN1(CORMN1,CORN1)
      CORMN2=AMIN1(CORMN2,CORN2)
      CORMX1=AMAX1(CORMX1,CORN1)
      CORMX2=AMAX1(CORMX2,CORN2)
10    PRINT 200,CORN(1,I),CORN(2,I),CORN1,CORN2,RLAT,RLNG
      IRI=CORMN1+CNTRX
      ICI=CORMN2+CNTRY
      IRF=CORMX1+CNTRX+1.
      ICF=CORMX2+CNTRY+1.
      RETURN
100   FORMAT(//' FILM COORDINATES AND PIXEL INCREMENTS(FROM CENTER) OF T
     .HE FOUR CORNERS OF THE RECTANGLE SPECIFIED')
200   FORMAT(1X4F10.3,10X2F10.3)
      END
```

```
      SUBROUTINE ERTXTS(NPIXS,IX,IY,NB,NEL,IBDS,NSAMPS,NSTRP,BANDS,
     .     FVECT,NT4PO)
      LOGICAL*1 BANDS,FVECT
      DIMENSION IBDS(NB),ISTRPS(4),NSAMPS(4),ISAMPI(4),ISAMPF(4)
      DIMENSION IPRTYI(4),IPRTYF(4),IBYTEI(4),IBYTEF(4)
      LOGICAL*1 IX(4000),IY(NEL,NB)
C     IF(BANDS.AND.FVECT) DIM IX(MAX0(4000,NEL*NB))
C
C     DEFINE FILE 90((IRF -IRI+1)*NB,NEL,L,IAV)
C     DEFINE FILE 90(9460,3240,L,IAV) WORKS FOR ALL ERTS CCT'S WITH LESS
C       THAN OR EQUAL TO 2340*3240 PIXELS.
C     DEFINE FILE 90(NRW90,NBYT90,L,IAV)
C     IF DEFINED FILE SPACE IS INSUFFICIENT, PROGRAM WILL RETURN NSTRP=0
C
      COMMON/COORD/IRI,IRF,ICI,ICF
      COMMON/DSK90/NRW90,NBYT90
      NSTRP=0
      DO 10 I=1,4
      IF(NPIXS*(I-1)+1.GT.ICF.OR.NPIXS*I.LT.ICI)GO TO 10
      NSTRP =NSTRP+1
      ISTRPS(NSTRP)=I
10    CONTINUE
      PRINT 1200,  (ISTRPS(I),I=1,NSTRP)
1200  FORMAT(' CCT STRIPS CONTAINING DESIRED DATA ARE'4I6)
C
C     FIND IF ALLOCATED DISK SPACE IS SUFFICIENT.  IF NOT PRINT ERROR
C     MESSAGE, SET NSTRP=0 AND RETURN.
C
      JREC=(IRF-IRI+1)*NB
      IF(JREC.LE.NRW90.AND.NEL.LE.NBYT90)GO TO 48
      PRINT 1000,  JREC,NEL,NRW90,NBYT90
1000  FORMAT(//' INSUFFICIENT FILE ALLOCATION FOR ERTXTS'/
     .           ' REQUIRED NRW90= 'I5/
     .           ' REQUIRED NBYT90='I5/
     .           ' SUPPLIED NRW90= 'I5/
     .           ' SUPPLIED NBYT90='I5)
      NSTRP=0
      RETURN
48    CONTINUE
C
C     SKIP IRI+1 RECORDS ON T[redacted] DESIRED STRIPS.
C
      DO 30 I=1,NSTRP
      IRI1=IRI+1
      IF(ISTRPS(I).EQ.1)IRI1=IRI-1
      IF(IRI1.EQ.0)GO TO 30
      DO 40 IREC=1,IRI1
40    CALL READNL(IX,IEND,LRECL,ISTRPS(I))
30    CONTINUE
C
C     FIND INITIAL AND FINAL SAMPLES TO BE EXTRACTED ON ISTRPS(I) FOR
```

```
C       I=1,NSTRP.
C
        DO 20 I=1,NSTRP
        ISAMPI(I)=MAX0(1,ICI-(ISTRPS(I)-1)*NPIXS)
        IPRTYI(I)=MOD(ISAMPI(I),2)
        IBYTEI(I)=(ISAMPI(I)-1)/2*8+2-IPRTYI(I)
C
        ISAMPF(I)=MIN0(NPIXS,ICF-(ISTRPS(I)-1)*NPIXS)
        IPRTYF(I)=MOD(ISAMPF(I),2)
        IBYTEF(I)=(ISAMPF(I)-1)/2*8-IPRTYF(I)+8
        NSAMPS(I)=ISAMPF(I)-ISAMPI(I)+1
        IF(I.GT.1)NSAMPS(I)=NSAMPS(I)+NSAMPS(I-1)
20      CONTINUE
C
C       EXTRACT AND COPY DATA ON DISK.
C
        JREC=1
        DO 70 IREC=IRI,IRF
C
C       READ DATA FROM EACH STRIP AND MOVE INTO ARRAY IY
C
        DO 80 I=1,NSTRP
        CALL READNL(IX,IEND,LRECL,ISTRPS(I))
        IF(BANDS)CALL ERTXT3(NB,NEL,IBYTEI,IBYTEF,IPRTYI,IBDS,NSAMPS,
             I,IX,IY)
        IF(.NOT.BANDS)CALL ERTXT4(NB,NEL,IBYTEI,IBYTEF,IPRTYI,IBDS,NSAMPS,
             I,IX,IY)
80      CONTINUE
C
C       ONE RECORD HAS BEEN FORMED IN ALL THE BANDS.  WRITE IT AS NB RECORDS
C       ON DISK.
C
        IF(.NOT.BANDS)GO TO 130
        DO 120 IB=1,NB
        CALL DAWN(90,JREC,IY(1,IB),NEL)
120     JREC=JREC+1
        IF(.NOT.FVECT)GO TO 70
        JEL=0
        DO 50 IEL=1,NEL
        DO 60 IB=1,NB
        JEL=JEL+1
60      IX(JEL)=IY(IEL,IB)
50      CONTINUE
        CALL SAWN(NTAPO,IX,NEL*NB)
        GO TO 70
130     CALL SAWN(NTAPO,IY,NEL*NB)
70      CONTINUE
C
C       END OF LOOP ON RECORDS
C
        RETURN
        END
```

```
      SUBROUTINE ERIXT3(NB,NEL,IBYTEI,IBYTEF,IPRTYI,IBDS,NSAMPS,I,IX,IY)
      DIMENSION IBYTEI(NB),IBYTEF(NB),IPRTYI(NB),IBDS(NB),NSAMPS(NB)
      LOGICAL*1 IX(1),IY(NEL,NB)
      DO 90 IB=1,NB
      IBYTE1=IBYTEI(I)+(IBDS(IB)-1)*2
      IBYTE2=IBYTEF(I)
      IF(I.EQ.1)JEL=1
      IF(I.GT.1)JEL=NSAMPS(I-1)+1
      IF(IPRTYI(I).EQ.1)GO TO 100
      IY(JEL,IB)=IX(IBYTE1)
      JEL=JEL+1
      IBYTE1=IBYTE1+7
100   CONTINUE
      DO 110 IEL=IBYTE1,IBYTE2,8
      IY(JEL,IB)=IX(IEL)
      JEL=JEL+1
      IY(JEL,IB)=IX(IEL+1)
110   JEL=JEL+1
90    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE FRTXT4(NB,NEL,IBYTEI,IBYTEF,IPRTYI,IBDS,NSAMPS,I,IX,IY)
      DIMENSION IBYTEI(NB),IBYTEF(NB),IPRTYI(NB),IBDS(NB),NSAMPS(NB)
      LOGICAL*1 IX(1),IY(NB,NEL)
      DO 90 IB=1,NB
      IBYTE1=IBYTEI(I)+(IBDS(IB)-1)*2
      IBYTE2=IBYTEF(I)
      IF(I.EQ.1)JEL=1
      IF(I.GT.1)JEL=NSAMPS(I-1)+1
      IF(IPRTYI(I).EQ.1)GO TO 100
      IY(IB,JEL)=IX(IBYTE1)
      JEL=JEL+1
      IBYTE1=IBYTE1+7
100   CONTINUE
      DO 110 IEL=IBYTE1,IBYTE2,8
      IY(IB,JEL)=IX(IEL)
      JEL=JEL+1
      IY(IB,JEL)=IX(IEL+1)
110   JEL=JEL+1
90    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE ERTXT6(IX,NB,NEL,NTAPO,HIST,IH,NREC)
      LOGICAL*1 IX(NEL),HIST
      DIMENSION IH(256,NB)
C
C     FORM INDIVIDUAL BAND IMAGES BY READING DATA FROM DISK WRITTEN BY
C     ERTXT5.  NB FILES OF OUTPUT ARE WRITEEN ON NTAPO.  IF HIST, GENERA
C     HISTOGRAMS OF EACH OF THE BANDS IN IH.
C
      IF(HIST)CALL SVSCI(IH,256*NB,0)
      DO 10 IB=1,NB
      WRITE(NTAPO)NREC,NEL
      JREC=IB
      DO 20 IREC=1,NREC
      READ(90'JREC)IX
      JREC=JREC+NB
      IF(HIST)CALL LRHSTG(IX,NEL,IH(1,IB))
20    IF(NTAPO.GT.0)WRITE(NTAPO)IX
10    IF(NTAPO.GT.0)END FILE NTAPO
      RETURN
      END
```

```
      SUBROUTINE LRHSTG(LX,N,IH)
      DIMENSION IH(256)
      LOGICAL*1 LX(N),LW(4)
      EQUIVALENCE(IW,LW(1))
      DATA IW/0/
      DO 10 I=1,N
      LW(4)=LX(I)
10    IH(IW+1)=IH(IW+1)+1
      RETURN
      END
```

```
      SUBROUTINE DARN(IDEV,IREC,X,N)
      LOGICAL*1 X(N)
      READ(IDEV'IREC)X
      RETURN
      ENTRY DAWN(IDEV,IREC,X,N)
      WRITE(IDEV'IREC)X
      RETURN
      ENTRY SARN(NTAPI,X,N)
      READ(NTAPI)X
      RETURN
      ENTRY SAWN(NTAPO,X,N)
      WRITE(NTAPO)X
      RETURN
      END
```

```
      SUBROUTINE ERTXT7(NPIXS,IX,IY,NBDS,IBDS,NELMIN,BANDS,FVECT,NTPFV)
      COMMON/COORD/IRI,IRF,ICI,ICF
      LOGICAL*1 IX(1),IY(4,1),BANDS,FVECT,SCRMBL
      DIMENSION IBDS(4)
      INTEGER*2 LLC
      INTEGER*2 LLA
      SCRMBL=NBDS.NE.4
      DO 5 I=1,NBDS
5     SCRMBL=SCRMBL.OR.IBDS(I).NE.I
C
C     DIMENSION IX, IY (4*3300)
      LLA=NPIXS*4
      NELMIN=ICF-ICI+1
C
C     SKIP TO IRI'TH RECORD OF DATA (NOTE: FIRST 2 RECORDS ARE ID AND
C     ANNOTATION: THEY ARE ASSUMED TO HAVE BEEN SKIPPED ON UNIT 1)
C
      DO 10 I=1,4
      IRI1=IRI+1
      IF(I.EQ.1)IRI1=IRI-1
      IF(IRI1.EQ.0)GO TO 10
      DO 20 IR=1,IRI1
20    CALL READNL(IX,IEND,LRECL,I)
10    CONTINUE
C
C     EXTRACT AND REARRANGE DATA FOR THE REGION  OF INTEREST.
C
      DO 30 IREC=IRI,IRF
C
C     MERGE DOUBLE BAND INTERLEAVED DATA.
C
      IAD=1
      DO 80 I=1,4
      CALL READNL(IX(IAD),IEND,LRECL,I)
      IF(I.EQ.1)CALL VMOV1(IX(LRECL-1),2,LLC)
80    IAD=IAD+LRECL-56
C
C     FIND IDEL=NO. OF GENUINE PIXELS BETWEEN SYNTHETIC PIXELS.
C
      IDEL=LLC/(LLA-LLC-6)
      CALL LINFIX(IX,IY,LLA,1,IDEL)
C
C     NOW IY HAS LLC FEATURE VECTORS.  FIND COLUMN ADDRESSES CORRESPONDIN
C     TO ICI,ICF( IT IS ASSUMED THAT ICI,ICF REFER TO THE FRAME INCLUDING
C     SYNTHETIC PIXELS).
C
      ICIO=ICI-(ICI-1)/(IDEL+1)
      ICFO=ICF-(ICF-1)/(IDEL+1)
      NELO=ICFO-ICIO+1
      NELMIN=MINO(NELMIN,NELO)
C
C     IF(BANDS)CALL ERTXT8 TO REARRANGE IY INTO BANDS AND WRITE ON DISK.
```

```
C     IF(.NOT.SCRMBL AND FVECT) WRITE FV'S ON NTPFV.
C     IF(SCRMBL AND FVECT) CALL ERTXT9 TO SCRAMBLE THE FV'S AND WRITE ON NTPFV.
C
      IF(BANDS)CALL ERTXT8(IY(1,ICIO),IX,NBDS,IBDS,NELO,IREC-IRI+1)
      IF(.NOT.SCRMBL.AND.FVECT)CALL SAWN(NTPFV,IY(1,ICIO),NELO*4)
      IF(SCRMBL.AND.FVECT)CALL ERTXT9(IY(1,ICIO),IX,NBDS,IBDS,NELO)
      IF(SCRMBL.AND.FVECT)CALL SAWN(NTPFV,IX,NELO*NBDS)
00    CONTINUE
      PRINT 100,NELMIN
100   FORMAT(' NUMBER OF PIXELS PER LINE AFTER REMOVAL OF SYNTHETIC PIXELS
     .LS='I5)
      RETURN
      END
```

```
LINFIX  CSECT
* THIS ROUTINE REARRANGES THE EIGHT BYTE ERTS PIXEL PAIRS TO
* SEPARATE THE ADJACENT PIXELS.  THE MAPPING IS
* 12345678 BECOMES 13572468
         USING *,12
         SAVE  (14,12),,*
         LR    12,15
         LR    11,13
         LA    13,SAVE
         ST    11,SAVE+4
         ST    13,8(11)
*
* LOAD PARAMETER LIST
*
         LM    2,6,0(1)     FETCH PARAMETER LIST
         ST    2,REC1       SAVE ADR OF INPUT ARRAY
         ST    3,LPIX       SAVE ADR OF OUTPUT ARRAY
         SR    0,0
         LH    0,0(4)       FETCH VALUE OF NPIXLN
         ST    0,NPIXLN     SAVE VALUE OF NPIXLN
         L     0,0(5)       LOAD LLC SWITCH VALUE
         ST    0,LLC        SAVE LLC 0=FALSE 1=TRUE
         L     0,0(6)       LOAD IDEL PIXEL REPEAT SPACING
         ST    0,IDEL
*
* SET UP INDICES FOR INNER LOOP
*
         L     2,REC1       ADR OF INPUT ARRAY
         L     3,LPIX       ADR OF OUTPUT ARRAY
         SR    4,4          ZERO OUT INDEX REGISTER
         SR    5,5          ZERO OUT TRANSITION REGISTER
         L     6,=F'8'      LOAD INCREMENT REGISTER
         L     7,NPIXLN     SET UP COMPARAND
         SLA   7,2          *  MULTIPLY BY 4
         S     7,=F'1'      *  NOW COMPARAND = NPIXLN*4 -1
         SR    8,8          COUNTER - UP TO REPT. PIXEL
         L     10,=F'1'     INCREMENT FOR BXH
         L     11,IDEL      COMPARAND FOR BXH - SPACING
          L    9,LLC        LOAD LINE LENGTH CORRECTION SWITCH
         LTR   9,9          TEST LINE LENGTH CORRECT SWITCH
         BNZ   LADJ         IF .TRUE. GO TO LINE ADJ CODING
*
* INNER LOOP - MAP BYTES INTO NEW POSITIONS
*
TOP      IC    5,0(4,2)     BAND 4 PIXEL LEFT
         STC   5,0(4,3)
         IC    5,1(4,2)          4       RIGHT
         STC   5,4(4,3)
         IC    5,2(4,2)          5       LEFT
         STC   5,1(4,3)
         IC    5,3(4,2)          5       RIGHT
         STC   5,5(4,3)
```

```
         IC     5,4(4,2)        6        LEFT
         STC    5,2(4,3)
         IC     5,5(4,2)        6        RIGHT
         STC    5,6(4,3)
         IC     5,6(4,2)        7        LEFT
         STC    5,3(4,3)
         IC     5,7(4,2)        7        RIGHT
         STC    5,7(4,3)
         BXLE   4,6,TOP
         B      END
LADJ     BXH    8,10,FIX1     COUNT PIXELS UP TO IDEL - REPEAT INTERVAL
         IC     5,0(4,2)
         STC    5,0(4,3)
         IC     5,2(4,2)
         STC    5,1(4,3)
         IC     5,4(4,2)
         STC    5,2(4,3)
         IC     5,6(4,2)
         STC    5,3(4,3)
PART2    BXH    8,10,FIX2
         IC     5,1(4,2)
         STC    5,4(4,3)
         IC     5,3(4,2)
         STC    5,5(4,3)
         IC     5,5(4,2)
         STC    5,6(4,3)
         IC     5,7(4,2)
         STC    5,7(4,3)
BOTTOM   BXLE   4,6,LADJ
         B      END
FIX1     SR     8,8
         S      3,=F'4'        ADJUST OUTPUT POINTER FOR DELETED PIXEL
         B      PART2          PROCESS RIGHT PIXEL OF THIS PAIR
FIX2     SR     8,8            RESET INDEX FOR INTERVAL COUNT
         S      3,=F'4'        RESET OUTPUT POINTER FOR DELETED PIXEL
         B      BOTTOM         END OF GROUP -
*
* END OR ROUTINE
*
END      L      13,SAVE+4
         RETURN (14,12),T,RC=0
SAVE     DS     18F
RFC1     DS     1F
LPIX     DS     1F
NPIXLN   DS     1F
LLC      DS     1F
IDEL     DS     1F
         END
```

```
      SUBROUTINE ERTXT8(IX,IY,NBDS,IBDS,NEL,IREC)
      LOGICAL*1 IX(4,NEL),IY(NEL)
      DIMENSION IBDS(NBDS)
      JREC=(IREC-1)*NBDS
      DO 10 I=1,NBDS
      DO 20 J=1,NEL
20    IY(J)=IX(IBDS(I),J)
      JREC=JREC+1
10    WRITE(90'JREC)IY
      RETURN
      END
```

```
      SUBROUTINE ERTXT9(IX,IY,NBDS,IBDS,NEL)
      LOGICAL*1 IX(4,NEL),IY(NBDS,NEL)
      DIMENSION IBDS(NBDS)
      DO 10 I=1,NBDS
      DO 10 J=1,NEL
10    IY(I,J)=IX(IBDS(I),J)
      RETURN
      END
```

```
      SUBROUTINE PRTHST(IH,N)
      DIMENSION IH(N)
      DO 10 I=1,N
      IF(IH(I).EQ.0)GO TO 10
      J=I-1
      WRITE(6,100)J,IH(I)
100   FORMAT(10XI4,18XI7)
10    CONTINUE
      RETURN
      END
```

5-2. COMPUTER CLASSIFICATION

1. NAME

EFFECT — Effective Figure of Merit Feature Selection Criterion

2. PURPOSE

This subroutine is used to implement a nonparametric feature selection criterion. The separability of classes of data from the remaining classes is required in the design of a sequential linear classifier.

3. CALLING SEQUENCE

CALL EFFECT (X, NS, CLASS, MOC, DE, NW1, NN, MM)

X	-	the array of data samples, with subscripts corresponding to feature number, class number, and sample number
NS	-	array containing number of data samples per class
CLASS	-	array containing class names (8 characters)
MOC	-	array containing class numbers in separability order
DE	-	array of interclass and intraclass distances for each feature
NW1	-	class counter
NN	-	number of features
MM	-	number of classes

4. INPUT/OUTPUT

4.1 Input

All input is via the arguments of the calling statement.

4.2 Output

Printed output consists of the values of the normalized figure of merit for each feature, for each class remaining under consideration. This is followed by a list of the effective figure of merit for each class, listed in descending order of merit.

5. EXITS

There are no nonstandard exits.

6. USAGE

Computer: IBM 360/65

Language: FORTRAN IV

7. EXTERNAL INTERFACES

7.1 System Subroutines

The subroutine SORTLS is called to arrange the effective figures of merit in descending order.

8. PERFORMANCE SPECIFICATIONS

8.1 Storage

Code:	EFFECT	3072	
	SORTLS	1684	
	Total	4756	bytes

8.2 Execution Time

In a typical case of 100 training samples for each of six classes of four band data, the interclass and intraclass distances (all elements of array DE) are computed in 9 seconds.

9. METHOD

The interclass and intraclass distances are labeled SUM 1 and SUM 2, respectively. They are computed for the I-th feature by the formulas:

$$\text{SUM 1 (I)} = \sum_{I3=1}^{MM} \sum_{I4=1}^{I3-1} \sum_{LK1=1}^{LKK1} \sum_{LK2=1}^{LKK2} \left[X(I, I3, LK1) - X(I, I4, LK2) \right]$$

$$\text{SUM 2 (I)} = \sum_{I3=1}^{MM} \sum_{LK1=1}^{LKK1} \sum_{LK2=1}^{LKK2} \left[X(I, I3, LK1) - X(I, I3, LK2) \right]$$

where MM is the number of classes and LKK1, LKK2 are the number of samples of the classes I3 and I4, respectively. The sums over sample numbers are the elements of array DE.

The normalized figure of merit is given by:

F(I) = SUM 1(I)/SUM 2(I)

The figures of merit for each feature are then combined to give the figure of merit for tne class.

10. COMMENTS

The elements of array DE are computed when the routine is called the first time (NW1 = 1). On succeeding calls, this calculation is bypassed.

11. LISTING

```
C
      SUBROUTINE EFFECT (X, NS, CLASS, MOC, DE, NW1, NN, MM)
C
C  EFFECTIVE FIGURE OF MERIT FEATURE SELECTION CRITERION
C
      DIMENSION X(NN,MM,1), NS(MM), MOC(MM), DE(MM,MM,NN), FC(20),
     .CFM(20)
      DOUBLE PRECISION CLASS(MM)
  142 FORMAT ('1'/20X,'EFFECTIVE FIGURES OF MERIT'/20X,26('*')/17X,16I7)
  143 FORMAT (I5,A9,5X,16F7.4/(19X,16F7.4))
  150 FORMAT (/20X,'COMBINED FIGURES OF MERIT'/20X,25('*')/)
  151 FORMAT (I21,A10,F14.5)
C
C COMPUTES INTER-CLASS AND INTRA-CLASS DISTANCES
C
      WRITE (6,142) (I, I=1,NN)
      IF (NW1.NE.1) GO TO 2000
      DO 1005 NF1=1,NN
      DO 2 I4=1,MM
      NC2 = NS(I4)
      DO 2 I5=1,I4
      NC3 = NS(I5)
      DE(I4,I5,NF1) = 0.0
      DO 3 LK2=1,NC2
      IF (I5.EQ.I4) NC3 = LK2 - 1
      DO 3 LK3=1,NC3
    3 DE(I4,I5,NF1) = DE(I4,I5,NF1) + ABS(X(NF1,I4,LK2) - X(NF1,I5,LK3))
    2 DE(I5,I4,NF1) = DE(I4,I5,NF1)
 1005 CONTINUE
      DO 1111 NC=1,MM
 1111 MOC(NC) = NC
C
C COMPUTES THE NORMALIZED FIGURE OF MERIT OF ALL REMAINING PATTERN
C CLASSES ALONG EACH OF THE FEATURE DIRECTIONS
C
 2000 CONTINUE
      DO 1100 J3=NW1,MM
      CFM(J3) = 1.0
      I3 = MOC(J3)
      AI3 = NS(I3)
      DO 3000 I=1,NN
      FCMIN = 1.0 E 50
      NST = 0
      SUM1 = 0.0
      SUM2 = 0.0
C
C  COMPUTE SUM1 - TOTAL OF INTERCLASS DISTANCES FROM CLASS I3 TO ALL
C  REMAINING CLASSES
C
      DO 6 J4=NW1,MM
      IF (J4.EQ.J3) GO TO 6
      I4 = MOC(J4)
      AI4 = NS(I4)
      NST = NST + NS(I4)
      SUM1 = SUM1 + DE(I3,I4,I)
C
C  COMPUTE SUM2 - TOTAL OF DISTANCES AMONG ALL REMAINING CLASSES,
C  EQUIVALENT TO INTRACLASS DISTANCE OF ALL REMAINING CLASSES CONSIDERED
C  AS ONE CLASS
C
```

```
      DO 7 J5=N+1,J4
      IF (J5.EQ.J3) GO TO 7
      I5 = MOC(J5)
      SUM2 = SUM2 + DE(I4,I5,I)
    7 CONTINUE
C
C  COMPUTE MINIMUM FIGURE OF MERIT FOR INDIVIDUAL CLASSES I3 AND I4
C
      S1 = DE(I3,I4,I) / (AI3*AI4)
      S2 = DE(I3,I3,I) / (AI3*(AI3-1.0)) + DE(I4,I4,I) / (AI4*(AI4-1.0))
      F = S1 / S2
      IF (F.LT.FCMIN) FCMIN = F
    6 CONTINUE
C
      ANST = NST
      SUM1 = SUM1 / (NS(I3)*NST)
      SUM2 = DE(I3,I3,I) / (AI3*(AI3-1.0)) + SUM2 / (ANST*(ANST-1.0))
      FC(I) = FCMIN * SUM1 / SUM2
      FC(I) = EXP(-1.0/FC(I))
C
C  COMPUTE CFM, COMBINED FIGURE OF MERIT, AND ORDER BY CFM TO DETERMINE
C  THE MOST SEPARABLE CLASS
C
 3000 CFM(J3) = CFM(J3) * FC(I)
      CFM(J3) = CFM(J3) ** (1.0/NN)
      WRITE (6,143) I3, CLASS(I3), (FC(NF), NF=1,NN)
 1100 CONTINUE
      CALL SORTLS (CFM, MOC, NW1, MM)
C
      PRINT 150
      DO 1251 NC1=N+1,MM
 1251 PRINT 151, MOC(NC1), CLASS(MOC(NC1)), CFM(NC1)
      CALL SORTSL (MOC, MOC, NW1+1, MM)
      RETURN
      END
```

12. TEST RESULTS

EFFECTIVE FIGURES OF MERIT

****

	1	2	3	4
1 URBAN 11	0.5182	0.5185	0.6611	0.6613
2 TRANS 15	0.4766	0.4914	0.5934	0.5568
3 AGRIC 21	0.4895	0.4442	0.7357	0.7526
4 DECID 31	0.5884	0.4832	0.4919	0.4551
5 EVGRN 32	0.7841	0.6479	0.5169	0.4727
6 WATER 61	0.4613	0.5555	0.9130	0.9263

COMBINED FIGURES OF MERIT

****

6	WATER 61	0.68228
5	EVGRN 32	0.59359
3	AGRIC 21	0.58904
1	URBAN 11	0.58543
2	TRANS 15	0.52743
4	DECID 31	0.50228

1. NAME

SNOPAL – Supervised Nonparametric Learning

2. PURPOSE

This subroutine is used to derive the coefficients of the linear functions used in a sequential linear classifier.

3. CALLING SEQUENCE

CALL SNOPAL (X, NS, CLASS, W, MOC, S, Y, B, NW1, NN, MM, NN1, MM1)

X	-	the array of training data, labeled by feature number, class number, sample number
NS	-	array containing number of data samples per class
CLASS	-	array containing class names (8 characters)
W	-	array of coefficients of the linear discriminant functions
MOC	-	array containing class numbers in order of testing
S	-	double precision array of dimension NN1 x NN1
Y	-	work array of length total number of training samples
B	-	same as Y
NW1	-	class counter
NN	-	number of features
MM	-	number of classes
NN1	-	NN + 1
MM1	-	MM - 1

4. INPUT/OUTPUT

4.1 Input

All input is via the calling arguments.

4.2 Output

The coefficients are output by the argument "W". Printed output is the coefficients and the errors for each iteration of the algorithm.

5. EXITS

There are no nonstandard exits.

6. USAGE

Computer: IBM 360/65

Language: FORTRAN IV

7. EXTERNAL INTERFACES

The subroutine GASINV is called to invert a matrix.

8. PERFORMANCE SPECIFICATIONS

8.1 Storage

SNOPAL	4172
GASINV	1838
Total	6010 bytes

8.2 Execution Time

The execution time for each call in constructing a sequential linear classifier varies because of variations in the number of iterations required and the number of training samples. In a typical four-band, six-class problem, the time spent by this routine approximately 1 minute.

9. METHOD

The method consists of maximizing the total distance of the training samples from the discriminant hyperplane, as described by J. C. Ho and R. L. Rashyap, "A Class of Iterative Procedures for Linear Inequalities," J. Siam on Control, 1966.

10. COMMENTS

The algorithm performs a maximum of 100 iterations. Otherwise, for four-band data, iterations cease when all coefficients change by less than 1 percent.

11. LISTING

```
C
      SUBROUTINE SNOPAL (X,NS,CLASS,W,MOC,S,Y,B,NW1,NN,MM,NN1,MM1)
C
C  SUPERVISED NON-PARAMETRIC LEARNING
C
      DIMENSION X(NN,MM,1), NS(MM), W(MM1,NN1), MOC(MM), Y(1), B(1)
      DOUBLE PRECISION CLASS(MM), S(NN1,NN1), DET
      LOGICAL TEST
      DATA NI /100/
  100 FORMAT (I8,I11,I8,4X,1P7E14.3/(31X,1P7E14.3))
  101 FORMAT (/I13,A10,10X,1P7E14.3/(33X,1P7E14.3))
  102 FORMAT (//22X,22('*')/22X,'* CLASS',I3,A10,' *'/22X,22('*')//' ITE
     .RATION NO. ',5X,'ERRORS',10X,'LINEAR DISCRIMINANT COEFFICIENTS'/
     .A22,'  OTHER'/)
  216 FORMAT ('1'/10X,'ORDERED CLASSES',20X,'ELEMENTS OF THE DISCRIMINAN
     .T VECTOR'/10X,15('*'),20X,35('*')/)
  220 FORMAT (/7X,'TOTAL ERRORS',I5)
C
C  INITIALIZE W, Y, AND B ARRAYS
C
      NW = MOC(NW1)
      NSW = NS(NW)
      NW2 = NW1 + 1
      DELTA = NN/400.0
      DO 1112 NFA=1,NN1
 1112 W(NW1,NFA) = 0.0
      NST2 = 0
      DO 1110 NC1=NW1,MM
      NC = MOC(NC1)
      NSC = NS(NC)
      DO 1110 NS1=1,NSC
      NST2 = NST2 + 1
      Y(NST2) =-1.0
 1110 B(NST2) = 1.0
C
C  COMPUTE INVERSE OF A(TRANSPOSE) A WHERE 'A' IS AUGMENTED MATRIX OF SAMPLES
C
      DO 130 I=1,NN
      DO 130 J=I,NN1
      S(I,J) = 0.0
      DO 131 NC1=NW1,MM
      NC = MOC(NC1)
      NSC = NS(NC)
      DO 131 NS1=1,NSC
      A1 = X(I,NC,NS1)
      IF (J.NE.NN1) A1 = A1*X(J,NC,NS1)
  131 S(I,J) = S(I,J) + A1
  130 S(J,I) = S(I,J)
      S(NN1,NN1) = NST2
      CALL GASINV (S, NN1, DET)
C
C  DO NI ITERATIONS OF THE HO-KASHYAP ALGORITHM, UNLESS ALL COEFFICIENTS
C  CHANGE BY LESS THAN DELTA = NN/4 PERCENT
C
      PRINT 102, NW, CLASS(NW), CLASS(NW)
      DO 1040 INDEX=1,NI
      TEST = .TRUE.
C
      DO 1101 I=1,NN1
      WO = W(NW1,I)
```

```
      DO 2101 J=1,NSW
      A2 = S(I,NN1)
      DO 140 K=1,NN
  140 A2 = A2 + S(I,K)*X(K,NW,J)
 2101 W(NW1,I) = W(NW1,I) + A2*ABS(Y(J))
      J = NSW
      DO 2000 NC1=NW2,MM
      NC = MOC(NC1)
      NSC = NS(NC)
      DO 2000 NS1=1,NSC
      J = J + 1
      A2 = S(I,NN1)
      DO 141 K=1,NN
  141 A2 = A2 + S(I,K)*X(K,NC,NS1)
 2000 W(NW1,I) = W(NW1,I) - A2*ABS(Y(J))
      IF (ABS(W(NW1,I)-W0).GT.ABS(DELTA*W0)) TEST = .FALSE.
 1101 CONTINUE
C
C  COMPUTE NEW DISCRIMINANT VALUES AND CLASSIFICATION ERRORS
C
      NERR1 = 0
      DO 1004 I=1,NSW
      IF (Y(I).GT.0.0) B(I) = B(I) + 2.0*Y(I)
      Y(I) = W(NW1,NN1)
      DO 1141 NF2=1,NN
 1141 Y(I) = Y(I) + W(NW1,NF2)*X(NF2,NW,I)
      IF (Y(I).LE.0.0) NERR1 = NERR1 + 1
 1004 Y(I) = Y(I) - B(I)
      I = NSW
      NERR2 = 0
      DO 1005 NC1=NW2,MM
      NC = MOC(NC1)
      NSC = NS(NC)
      DO 1005 NS1=1,NSC
      I = I + 1
      IF (Y(I).GT.0.0) B(I) = B(I) + 2.0*Y(I)
      Y(I) = W(NW1,NN1)
      DO 145 NF2=1,NN
  145 Y(I) = Y(I) + W(NW1,NF2)*X(NF2,NC,NS1)
      IF (Y(I).GT.0.0) NERR2 = NERR2 + 1
 1005 Y(I) = -Y(I) - B(I)
C
      PRINT 100, INDEX, NERR1, NERR2, (W(NW1,NFA), NFA=1,NN1)
      IF (TEST) GO TO 1010
 1040 CONTINUE
 1010 NERR = NERR1 + NERR2
      PRINT 220, NERR
C
      IF (NW1.NE.MM1) RETURN
      WRITE (6,216)
      DO 1220 NW=1,MM1
      NC = MOC(NW)
 1220 PRINT 101, NC, CLASS(NC), (W(NW,NFA), NFA=1,NN1)
      PRINT 101, MOC(MM), CLASS(MOC(MM))
      RETURN
      END
```

12. TEST OUTPUT

• CLASS 4 DECID 31 •

ITERATION NO.	ERRORS DECID 31	ERRORS OTHER	LINEAR DISCRIMINANT COEFFICIENTS				
1	1	11	-1.342E-01	5.159E-02	-8.379E-02	7.075E-02	4.488E 00
2	1	10	-1.825E-01	7.286E-02	-1.107E-01	1.024E-01	5.666E 00
3	1	10	-2.065E-01	8.120E-02	[illegible]	1.176E-01	6.374E 00
4	1	10	-2.286E-01	9.041E-02	-1.349E-01	1.292E-01	6.929E 00
5	2	10	-2.433E-01	9.683E-02	-1.428E-01	1.379E-01	7.318E 00
6	2	10	-2.579E-01	1.026E-01	-1.478E-01	1.433E-01	7.676E 00
7	2	7	-2.690E-01	1.070E-01	-1.519E-01	1.473E-01	7.962E 00
8	2	6	-2.793E-01	1.114E-01	-1.562E-01	1.516E-01	8.223E 00
9	2	6	-2.879E-01	1.149E-01	-1.601E-01	1.555E-01	8.458E 00
10	2	6	-2.958E-01	1.186E-01	-1.643E-01	1.599E-01	8.661E 00
11	2	6	-3.032E-01	1.219E-01	-1.681E-01	1.637E-01	8.855E 00
12	2	5	-3.097E-01	1.250E-01	-1.720E-01	1.675E-01	9.031E 00
13	2	5	-3.158E-01	1.277E-01	-1.755E-01	1.710E-01	9.198E 00
14	2	5	-3.213E-01	1.302E-01	-1.787E-01	1.740E-01	9.355E 00
15	2	5	-3.265E-01	1.324E-01	-1.817E-01	1.767E-01	9.505E 00
16	2	5	-3.314E-01	1.344E-01	-1.843E-01	1.791E-01	9.647E 00
17	2	5	-3.361E-01	1.363E-01	-1.868E-01	1.813E-01	[illegible]
18	2	5	-3.405E-01	1.381E-01	-1.892E-01	1.834E-01	9.913E 00
19	2	5	-3.446E-01	1.398E-01	-1.914E-01	1.854E-01	[illegible]
20	2	5	-3.486E-01	1.414E-01	-1.936E-01	1.874E-01	1.015E 01
21	2	5	-3.523E-01	1.428E-01	-1.958E-01	1.893E-01	1.027E 01
22	2	5	-3.559E-01	1.443E-01	-1.979E-01	1.913E-01	1.037E 01
23	2	4	-3.593E-01	1.457E-01	-2.000E-01	1.932E-01	1.049E 01
24	2	4	-3.625E-01	1.470E-01	-2.020E-01	1.951E-01	1.057E 01

TOTAL ERRORS 6

1. NAME

NOPACA – Nonparametric Classification Algorithm

2. PURPOSE

This subroutine has as its purpose the implementation of this method of classification.

3. CALLING SEQUENCE

CALL NOPACA (X, NW, W, MOC, NSS, NN, NN1, MM1)

X	-	array of feature vectors to be classified
NW	-	array of class numbers assigned to input feature vectors
W	-	array of coefficients of the linear discriminant functions
MOC	-	array containing class numbers in order of testing
NSS	-	number of feature vectors to be classified
NN	-	number of spectral bands
NN1	-	NN + 1
MM1	-	number of classes less one

4. INPUT/OUTPUT

4.1 Input

All input is via the items in the calling statement.

4.2 Output

The output is described under CALLING SEQUENCE.

5. EXITS

There are no nonstandard exits.

6. USAGE

Computer: IBM 360/65

Language: FORTRAN IV

7. EXTERNAL INTERFACES

None.

8. PERFORMANCE SPECIFICATIONS

8.1 Storage

792 bytes

8.2 Execution Times

The time required to classify a four-band feature vector is 0.06 millisecond per class present.

9. METHOD

The unknown feature vector is used to evaluate the discriminant functions in the order of class separability determined by subroutine EFFECT. The unknown feature vector is assigned to that class for which the evaluation is positive.

10. COMMENTS

None.

11. LISTING

```
C
      SUBROUTINE NORACA (X, NW, W, MOC, NSS, NN, NN1, MM1)
C
C NON-PARAMETRIC CLASSIFICATION OF A STRING OF NSS FEATURE VECTORS
C USING PRE-LEARNED LINEAR DISCRIMINANT FUNCTIONS
C
      DIMENSION X(NN,NSS), NW(NSS), W(MM1,1), MOC(1)
C
      DO 20 NS1=1,NSS
      DO 1 NW1=1,MM1
      G = W(NW1,NN1)
      DO 2 NF1=1,NN
    2 G = G + W(NW1,NF1) * X(NF1,NS1)
      IF (G.GT.0.0) GO TO 3
    1 CONTINUE
      NW(NS1) = MOC(MM1+1)
      GO TO 20
    3 NW(NS1) = MOC(NW1)
   20 CONTINUE
      RETURN
      END
```

5-3. GEOGRAPHIC REFERENCING

1. NAME

GEOGREF

2. PURPOSE

Find a geometric transformation from one coordinate system to another such that the mean squared error at a given set of control points is minimized. The transformation is linear, accounting for rotation, scale change, skew, and translation.

3. CALLING SEQUENCE

This is a main program. It is currently on a partitioned data set as an executable module.

4. INPUT-OUTPUT

4.1 Input

The following input parameters should be supplied in data cards according to the formats and read statements indicated below.

```
1     READ  100,NCP,ICI,ISP,NTRY
      IF(NCP.LE.0)STOP
         .
         .
         .
      DO 10 N=1,NCP
      N2=N*2
      READ 102,X(N2-1),X(N2),Y(N2-1),Y(N2),TITL
         .
         .
         .
10    CONTINUE
         .
         .
         .
      GO TO 1
100   FORMAT (4I6)
102   FORMAT (4D12.0,32A1)
```

where

NCP=Number of control points (if NCP≤0, the program stops)
ICI, ISP are special parameters to be used when handling Landsat

images, with the ground control points' coordinates measured relative to the image without the synthetic pixels removed and the transformation required being from the image with no synthetic pixels. ICI is then equal to the initial pixel number on the Landsat frame starting from which the region of interest was extracted and ISP is the number of real pixels between synthetic pixels. If ISP is specified as zero, this special case is ignored and no corrections are applied to the coordinates (Y(N2)). It is generally more convenient to remove synthetic pixels in advance and supply ISP=0. NTRY=Number of fits to be found for the current set of control points using the successive elimination procedure (See Section 9).

X (N2 - 1), X(N2): coordinates of the N'th control point in the reference image (e.g. UTM coordinates)

Y (N2-1), Y(N2): coordinates of N'th control point in the observed image (e.g. Landsat pixel coordinates).

The transformation is found from the X's to the Y's. Note that the loop starting at statement number 1 indicates that the program finds transformations for several sets of control points, until terminated by, say, a blank card while reading NCP.

TITLE: Arbitrary 32 character title.

4.2 Output

The output of this program is a printout of the control point coordinates, the fit parameters found and a table of errors at all the control points. A typical output is attached at the end.

4.3 File Storage

None.

5. EXITS

Not applicable.

6. USAGE

The program is in FORTRAN IV and implemented on the IBM360 using the H compiler. It is in the user's library in its executable form.

7. EXTERNAL INTERFACES

The linkage with the subroutines required by this program is shown in the following table.

Calling Program	Programs Called
GEOGREF	RSVP EHVFIT SUBRT
EHVFIT	SORT DPMMV LNLLS SUBRT
LNLLS	GAUSS SUBRT
SORT	MVMRMR
GAUSS	SUBRT BIORTH
BIORTH	SCLR DOT

8. PERFORMANCE SPECIFICATIONS

8.1 Storage

The program is 7688 bytes long, but including external references required and the buffers, this program requires 62K bytes of storage.

8.2 Execution Time

Depends on the number of cases to be considered and NCP,NTRY in each case. With NCP=36 and NTRY=6, this program takes approximately four seconds per case.

8.3 I/O Load

None

8.4 Restrictions

None

9. METHOD

In the special case of Landsat data (described in Section 4.1) the routine RSVP is first used to modify the coordinates Y(N*2) for N=1, ...,NCP. The means of both X and Y coordinates over N=1,..., NCP are found and subtracted in order avoid possible inversion of a matrix with large numbers during the determination of the transformation.

The routine EHVFIT is used to determine the transformation. This routine considers a given subset of the control points supplied and, using the least squares fit program LNLLS [20], finds the goemetric transformation parameters. The transformation so formed is used to compute the error at all the control points used for the fit. The mean and variance of the error are found. The points with error greater than or equal to the (mean + variance) are included in the set of points to be eliminated for the next trial.

The main program GEOGREF calls EHVFIT NTRY times. During the first call the set of points to be eliminated is null. During the subsequent calls this set is appended with points causing high error so that with each attempt the RMS error at the fit points is reduced. After each call to EHVFIT, the set of points used for finding the transformation, the RMS error at the fit points, the values of the parameters defining the transformation, a table of errors at all the control points, the overall RMS error and the set of points to be omitted next are printed.

10. COMMENTS

The details of the subroutines are omitted here. The least squares fit routine LNLLS is described elsewhere [20].

11. LISTINGS

The listings of the program, the associated subroutines are attached at the end.

12. TESTS

The program has been tested for several sets of control points.

```
C
C       TO FIND GEOMETRIC TRANSFORMATION NEEDED FOR GEOGRAPHIC REFERENCING
C
        DIMENSION W(100), ICOMB(50), ICOMBO(50)
        DOUBLE PRECISION NORM, X(100), Y(100), XP(100), YP(100), XO(100),
       .YO(100), W1(12), GP1(6), TB(6), DX, DY
        LOGICAL*1 TITL(32)
C
C       D'N W(2*NCP), ICOMB(NCP), ICOMBO(NCP)
C       D.P. X(2*NCP), Y(2*NCP), XP(2*NCP), YP(2*NCP), XO(2*NCP), Y(2*NCP)
C       WHERE NCP IS THE MAXIMUM NUMBER OF G. C. P.'S EXPECTED FOR ONE FIT.
        COMMON /NORN/ NORM /PRINV/ INV
        INV=0
        RADDEG = 180.0/3.14159265
1       CONTINUE
        READ 100,NCP,ICI,ISP,NTRY
C
C       NCP IS THE NUMBER OF CONTROL POINTS TO BE USED IN THE FIT.
C
        IF(NCP.LE.0)STOP
        WRITE(6,101)
        XM1=0.
        XM2=0.
        YM1=0.
        YM2=0.
        DO 10 N=1,NCP
        N2=2*N
C
C       READ E,N ROW, COL.
C
        READ 102,X(N2-1),X(N2),Y(N2-1),Y(N2),TITL
        NEWY=Y(N2)
        CALL WSVP(NEWY,ISP,ICI,NEWY)
        Y(N2)=NEWY
        XM1=XM1+X(N2-1)
        XM2=XM2+X(N2)
        YM1=YM1+Y(N2-1)
        YM2=YM2+Y(N2)
10      PRINT 103,N,X(N2-1),X(N2),Y(N2-1),Y(N2),TITL
        PRINT 105,XM1,XM2,YM1,YM2
        XM1=XM1/NCP
        XM2=XM2/NCP
        YM1=YM1/NCP
        YM2=YM2/NCP
        PRINT 104,XM1,XM2,YM1,YM2
        PRINT 101
        DO 15 N=1,NCP
        N2=N*2
        X(N2-1)=X(N2-1)-XM1
        X(N2)=X(N2)-XM2
        Y(N2-1)=Y(N2-1)-YM1
```

```
      Y(N2)=Y(N2)-YM2
15    PRINT 103,N,X(N2-1),X(N2),Y(N2-1),Y(N2)
C
      W(1)=-1.
      MCP=NCP
      K=0
      NERR=0
      DO 1000 KK=1,NTRY
      NCPK=NCP-K
      CALL EHVFIT(K, ICOMB, NCP, X, Y, XP, YP, W, GP1, NERR, ERMIN,
     1     ERMEAN, ERVAR, ICOMBO)
      IF(NERR.EQ.1)GO TO 20
C
C     ERROR CONDITION -- NO LEAST SQUARES FIT.
C
      PRINT 106
      STOP
C
20    CONTINUE
      IF(ERMIN.GT.0.)ERMIN=SQRT(ERMIN/NCPK)
      DX = GP1(5) + YM1 - GP1(1)*XM1 - GP1(2)*XM2
      DY = GP1(6) + YM2 - GP1(3)*XM1 - GP1(4)*XM2
      WRITE(6,121)NERR
      WRITE(6,124)NCPK
      WRITE(6,122) (ICOMBO(J), J=1, NCPK)
      WRITE(6,123) ERMIN, ERMEAN, ERVAR
      WRITE(6,108)NORM
      WRITE(6,107)(GP1(J),J=1,4),DX,DY
C
      WRITE(6,111)
      RMS=0.
      DO 25 N=1,NCP
      N2=2*N
C
      CALL SUBRT (N,X,GP1,W1,TB)
C
      DU=Y(N2-1)-TB(1)
      DV=Y(N2)-TB(2)
      SQ=DU**2+DV**2
      RMS=RMS+SQ
      XMAG=SQRT(SQ)
      XDIR=RADDEG*ATAN2(DV,DU)
      PRINT 109,N,Y(N2-1),Y(N2),TB(1),TB(2),XMAG,YDIR,DU,DV
25    CONTINUE
      RMS=SQRT(RMS/FLOAT(NCP))
      WRITE(6,114)RMS
      WRITE(6,119)K
      IF(K.NE.0)WRITE(6,120) (ICOMB(J),J=1,K)
      IF(NCP-K.LE.2)GO TO 1
1000  CONTINUE
      GO TO 1
C
C     FORMAT STATEMENTS.
C
100   FORMAT(12I6)
```

```
101    FORMAT('1',39X,'CONTROL POINTS'/28X,'GEOGRAPHIC',21X,'PIXEL'/28X,
      .'COORDINATES',18X,'COORDINATES')
102    FORMAT(4D12.0,32A1)
103    FORMAT(/I5,2(6X,1PD11.4,',',D11.4),3X56A1)
104    FORMAT('0MEANS OF INPUT DATA ARE'4E20.8/
      .       '0THESE MEANS ARE FIRST SUBTRACTED')
105    FORMAT('0 SUMS OF INPUT DATA ARE'4E20.8)
106    FORMAT('1  LEAST SQUARES FIT NOT ACHIEVED')
107    FORMAT(///15H FIT PARAMETERS/8HOA(11) =,1PE15.8/8HOA(12)= ,
      .       E15.8/8HOA(21) =,E15.8/8HOA(22) = ,E15.8/5HODX= ,E15.8/
              5HODY= ,E15.8)
108    FORMAT('0      NORM ='1PD11.4)
109    FORMAT(/1X,I2,4(2F10.2,8X))
111    FORMAT(///20X43HCOMPARISON OF OBSERVED AND PREDICTED VALUES//
              9X8HOBSERVED,20X9HPREDICTED,22X5HERROR/
      .       7X11HCOORDINATES,18X11HCOORDINATES,13X9HMAGNITUDE,
              2X9HDIRECTION,11X7HX ERROR,3X7HY ERROR)
112    FORMAT(1H0,1PD11.4,1H,,D11.4,6XD11.4,1H,,D11.4,6X2F11.4)
114    FORMAT(///12H RMS ERROR =,1PE11.4)
119    FORMAT('0 NUMBER OF DATA POINTS TO BE OMITTED NEXT='I3)
120    FORMAT('0 SET OF POINTS TO BE OMITTED'/(1X30I4))
121    FORMAT('1  LEAST SQUARES FIT ACHIEVED, NERR =',I2)
122    FORMAT(' SET OF POINTS USED IN COMPUTING THE FIT PARAMETERS:'/
              (1X30I4))
123    FORMAT(//' ERRORS OVER THE SET OF POINTS USED FOR THE FIT'/
      .       ' RMS ERROR='1PE11.4,'; MEAN ERROR='1PE11.4,'; STANDARD DEVIA
       TION='1PE11.4,'.')
124    FORMAT(' NUMBER OF POINTS USED FOR THE FIT='I3)
       END
```

```
      SUBROUTINE EHVFIT(K,ICOMB,N,X,Y,XP,YP,W,GP,NERR, ERR, ERMEAN,
     *     ERVAR, ICOMBO)
C
C     PERFORM FITS BY ELIMINATING POINTS WITH HIGH VARIANCE.
C     FIRST, FIND THE FIT AND ERRORS AT THE POINTS SUPPLIED.
C     NEXT, IDENTIFY LOCATIONS OF HIGH ERROR AND STORE THEIR INDICES IN
C     ICOMB IN PREPARATION FOR THE NEXT CALL OF EHVFIT.
C
      DIMENSION ICOMB(N),W(2,N),ICOMBO(N)
      REAL*8 X(2,N),Y(2,N),XP(2,N),YP(2,N),GP(6),T(6),W1(12)
      NK=N-K
      CALL DPMMV(X,XP,2,N,ICOMB,K)
      CALL DPMMV(Y,YP,2,N,ICOMB,K)
      CALL LNLLS(W,XP,YP,GP,NERR,6,NK,2,0,0,4HLIN.,T)
      IF(NERR.NE.1)RETURN
C
C     FIND MEAN AND VARIANCE OF ERROR.
C
      ERMEAN=0
      FRVAR=0
      DO 50 J=1,NK
      CALL SUBRT(J,XP,GP,W1,T)
      DU=YP(1,J)-T(1)
      DV=YP(2,J)-T(2)
      ERR=DU**2+DV**2
      XP(1,J)=SQRT(ERR)
      ERVAR=ERVAR+ERR
50    ERMEAN=ERMEAN+XP(1,J)
      ERMEAN=ERMEAN/NK
      ERR=ERVAR
      ERVAR=ERVAR/NK-ERMEAN**2
      IF(ERVAR.GT.0)ERVAR=SQRT(ERVAR)
      ERTHR=ERMEAN+ERVAR
C
C     FIND THE SET OF POINTS TO BE ELIMINATED.
C     FIRST, SET ICOMBO=(1,2,...,N)-ICOMB
C
      I=1
      L=0
      DO 10 J=1,N
      IF(I.GT.K)GO TO 40
      IF(J.EQ.ICOMB(I))GO TO 20
40    L=L+1
      ICOMBO(L)=J
      GO TO 10
20    I=I+1
10    CONTINUE
C
C     NOW, ICOMBO(L) IS THE INDEX IN THE X SET CORRESPONDING TO L IN THE
C     XP SET.
C
```

```
      L=K
      DO 30 J=1,NK
      IF(XP(1,J).LT.ERTHR)GO TO 30
      L=L+1
      ICOMB(L)=ICOMBO(J)
   30 CONTINUE
      K=L
      CALL SORT(ICOMB,1,K,K,1,P,PP)
      RETURN
      END
```

```
      SUBROUTINE SORT(A,II,JJ,MM,NN,T,TT)
      DIMENSION A(MM,NN),T(NN),TT(NN),IU(16),IL(16)
      INTEGER A,T,TT
      M=1
      I=II
      J=JJ
5     IF(I.GE.J)GO TO 70
10    K=I
      IJ=(I+J)/2
      CALL MVMRMR(A,MM,NN,T,1,IJ,1)
      IF(A(I,1).LE.T(1))GO TO 20
      CALL MVMRMR(A,MM,NN,A,MM,I,IJ)
      CALL MVMRMR(T,1,NN,A,MM,1,I)
      CALL MVMRMR(A,MM,NN,T,1,IJ,1)
20    L=J
      IF(A(J,1).GE.T(1))GO TO 40
      CALL MVMRMR(A,MM,NN,A,MM,J,IJ)
      CALL MVMRMR(T,1,NN,A,MM,1,J)
      CALL MVMRMR(A,MM,NN,T,1,IJ,1)
      IF(A(I,1).LE.T(1))GO TO 40
      CALL MVMRMR(A,MM,NN,A,MM,I,IJ)
      CALL MVMRMR(T,1,NN,A,MM,1,I)
      CALL MVMRMR(A,MM,NN,T,1,IJ,1)
      GO TO 40
30    CALL MVMRMR(A,MM,NN,A,MM,K,L)
      CALL MVMRMR(TT,1,NN,A,MM,1,K)
40    L=L-1
      IF(A(L,1).GT.T(1))GO TO 40
      CALL MVMRMR(A,MM,NN,TT,1,L,1)
50    K=K+1
      IF(A(K,1).LT.T(1))GO TO 50
      IF(K.LE.L)GO TO 30
      IF(L-I.LE.J-K)GO TO 60
      IL(M)=I
      IU(M)=L
      I=K
      M=M+1
      GO TO 80
60    IL(M)=K
      IU(M)=J
      J= L
      M=M+1
      GO TO 80
70    M=M-1
      IF(M.EQ.0)RETURN
      I=IL(M)
      J=IU(M)
80    IF(J-I.GE.II)GO TO 10
      IF(I.EQ.II)GO TO 5
      I=I-1
90    I=I+1
```

```
      IF(I.EQ.J)GO TO 70
      CALL MVMRMR(A,MM,NN,T,1,I+1,1)
      IF(A(I,1).LE.T(1))GO TO 90
      K=I
100   CALL MVMRMR(A,MM,NN,A,MM,K,K+1)
      K=K-1
      IF(T(1).LT.A(K,1))GO TO 100
      CALL MVMRMR(T,1,NN,A,MM,1,K+1)
      GO TO 90
      END
```

```
      SUBROUTINE MVMRMR(A,MA,N,B,MB,IA,IB)
      DIMENSION A(MA,N),B(MB,N)
      DO 10 J=1,N
10    B(IB,J)=A(IA,J)
      RETURN
      END
```

```
C
      SUBROUTINE DPMMV%X,Y,M,N,ICOMB,K<
C
C     TO MOVE PART OF A DOUBLE PRECISION MATRIX SPECIFIED BY INDICES
C     BETWEEN 1 AND N OTHER THAN ICOMB%1<,...., ICOMB%K< INTO D. P.
C     MATRIX Y.
C
      DIMENSION ICOMB%K<
      DOUBLE PRECISION X%M,N<,Y%M,N<
C
      I#1
      L#1
      DO 10 J#1,N
      IF%I.GT.K<GO TO 40
      IF%J.EQ.ICOMB%I<<GO TO 20
40    DO 30 JJ#1,M
30    Y%JJ,L<#X%JJ,J<
      L#L&1
      GO TO 10
20    I#I&1
10    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE LNLLS (W, X, Y, GP1, NERR, NP, NDP, NDIM, EPS,
     .                  NIT, LINNON, TB)
C
C     PROGRAM TO CALL GAUSS AND MANAGE ITERATIVE CALCULATION
C      FOR NONLINEAR PROBLEMS
C
C
      DOUBLE PRECISION X(1), Y(1), GP1(1), TB(NP), XX(30,30)
      DIMENSION W(1)
      DATA LINEAR /4HLIN.  /
      COMMON /PRINV/ INV
C
      IF (W(1) .GT. 0.) GO TO 10
C     SET ALL WEIGHTS TO UNITY IF NONE WERE SUPPLIED
      IUP = NDP * NDIM
      DO 5 I=1,IUP
  5   W(I) = 1.0
 10   CONTINUE
C     HANDLE LINEAR AND NONLINEAR PROBLEMS DIFFERENTLY
      IF (LINNON .EQ. LINEAR) GO TO 60
C     NONLINEAR PROBLEM --
C         SET IMODE = 2, AND START ITERATIVE PROCEDURE
      IMODE = 2
 15   DO 30 N=1,NIT
C
      CALL GAUSS (W, X, Y, GP1, NERR, NP, NDP, NDIM, XX, IMODE, TB)

      IF (NERR .EQ. 2) GO TO 70
      IF (IMODE .EQ. 1) GO TO 35
C     TEST FOR CONVERGENCE
      DO 20 I=1,NP
      IF (GP1(I)) 16, 17, 16
 16   CONV = DABS (TB(I) / GP1(I))
      GO TO 18
 17   CONV = DABS (TB(I))
 18   IF (CONV .GT. EPS) GO TO 25
 20   CONTINUE
      GO TO 35
C     CONVERGENCE NOT YET ATTAINED
C     UPDATE GP1, CONTINUE ITERATING
 25   DO 30 I=1,NP
 30   GP1(I) = GP1(I) + TB(I)
C     CONVERGENCE WAS NOT ATTAINED WITHIN SPECIFIED NUMBER OF
C      ITERATIONS
      NERR = 3
      WRITE (6,100) NIT, EPS
 100  FORMAT (20HODID NOT CONVERGE IN,I4,
     .          28H ITERATIONS WITH CRITERION,E15.8)
      GO TO 45
C     CONVERGENCE WAS ACHIEVED (OR PROBLEM IS LINEAR)
```

```
C       MAKE FINAL UPDATES OF GP1
 35     DO 40 I=1,NP
 40     GP1(I) = GP1(I) + TB(I)
C       PRINT INVERSE MATRIX
 45     IF (INV .LE. 0) GO TO 55
        WRITE (6,101)
 101    FORMAT (//20H0      INVERSE MATRIX)
        DO 50 I=1,NP
 50     WRITE (6,102) (XX(I,J),J=1,NP)
 102    FORMAT (1H0,1P8D14.6)
 55     RETURN
C       SET UP FOR LINEAR PROBLEM
 60     IMODE = 1
        DO 65 I=1,NP
 65     GP1(I) = 0.
        GO TO 15
C       NORMAL MATRIX WAS SINGULAR
 70     WRITE (6,103)
 103    FORMAT (16H0SINGULAR MATRIX)
        RETURN
        END
```

```
      SUBROUTINE GAUSS (W, X, Y, GP1, NERR, NP, NDP, NDIM, XX,
     .                      IMODE, TB)
C
C  VECTOR LEAST SQUARES SUBROUTINE
C
C
      DOUBLE PRECISION X(1), Y(1), GP1(1), TB(1)
      DOUBLE PRECISION XX(30,30)
      DOUBLE PRECISION A(30,30), B(30), DERIV(60), NORM, FI(2), YC(2)
      DIMENSION W(1)
C
C THE FIRST NDIM ELEMENTS OF W AND Y ARE FOR THE FIRST OBSERVATION,
C   THE NEXT NDIM ARE FOR THE SECOND, ETC.  IF THE INDEPENDENT
C   VARIABLE IS A VECTOR, A SIMILAR CONVENTION IS OBSERVED FOR X.
C
C CURRENT DIMENSIONS ARE FOR NDIM = 2.  FOR OBSERVATIONS WITH
C   HIGHER DIMENSIONALITY, THE DIMENSIONS OF FI, YC, AND (PERHAPS)
C   DERIV MUST BE INCREASED.
C
      COMMON /NORN/ NORM
C
C
C ASSUME NO ERROR INITIALLY
C
      NERR = 1
C
C INITIALIZE A AND B MATRICES
C
      DO 110 L=1,NP
      B(L) = 0.
      DO 110 M=1,NP
 110  A(L,M) = 0.
C
C FOR EACH OBSERVATION DETERMINE (OLD) ESTIMATES AND DERIVATIVES.
C
C FORM SUMS TO GIVE MATRIX ELEMENTS.
C
      DO 111 J=1,NDP
C
      CALL SUBRT (J, X, GP1, DERIV, YC)
C
C THE FIRST NDIM ELEMENTS OF DERIV ARE PARTIAL DERIVATIVES OF
C   SUCCESSIVE VECTOR COMPONENTS WITH RESPECT TO THE FIRST FIT
C   PARAMETER, THE SECOND NDIM ARE PARTIALS WITH RESPECT TO THE
C   SECOND, ETC.
C
      GO TO (1111, 1113), IMODE
C
C DIRECT CALCULATION OF PARAMETERS
C
 1111 DO 1112 I=1,NDIM
      ISUB = (J - 1) * NDIM  +  I
```

```
1112 FI(I) = Y(ISUB)
     GO TO 1115
C
C  ITERATIVE CALCULATION OF PARAMETERS
C
1113 DO 1114 I=1,NDIM
     ISUB = (J - 1) * NDIM  +  I
1114 FI(I) = Y(ISUB) - YC(I)
1115 CONTINUE
     KSUB = 0
     DO 111 K=1,NP
     LSUB = (J - 1) * NDIM
     DO 111 L=1,NDIM
     KSUB = KSUB + 1
     LSUB = LSUB + 1
     B(K) = B(K)  +  DERIV(KSUB) * W(LSUB) * FI(L)
     MSUB = KSUB - NDIM
     DO 111 M=K,NP
     MSUB = MSUB + NDIM
 111 A(K,M) = A(K,M)  +  DERIV(KSUB) * W(LSUB) * DERIV(MSUB)
C
C COMPLETE NORMAL MATRIX (LOWER LEFT TRIANGLE)
C
     DO 112 M=2,NP
     K = M - 1
     DO 112 I=1,K
 112 A(M,I) = A(I,M)
C
C  INVERT A
C
     CALL BIORTH (A, XX, NP)
C
C CHECK NORM AND SET NERR ACCORDINGLY.
C
     IF (NORM .LT. 1.D-08) GO TO 114
     NERR = 2
C
     RETURN
C
C COMPUTE ESTIMATE OF (CORRECTIONS TO) FIT PARAMETERS, IF MATRIX
C  INVERSION WAS SUCCESSFUL.
C
 114 DO 115 I=1,NP
     TB(I) = 0.
     DO 115 J=1,NP
 115 TB(I) = XX(I,J) * B(J)  +  TB(I)
C
C FOR ITERATIVE SOLUTION, THIS IS AN INCREMENT TO BE ADDED TO
C  THE PREVIOUS ESTIMATE.
C
     RETURN
C
     END
```

```
      SUBROUTINE SUBRT (J, X, GP1, DERIV, YC)
C     PROGRAM TO GENERATE TRANSFORMED VECTORS AND THEIR PARTIAL .
C        DERIVATIVES, AS FUNCTIONS OF THE (FIT) PARAMETERS OF
C        THE TRANSFORMATION
C     TRANSFORMATION INCLUDES ROTATION AND SKEW
C        (SCALE FACTORS ARE IMPLICIT IN THE COEFFICIENTS)
C
C
      DOUBLE PRECISION X(1), GP1(1), DERIV(1), YC(1)
C
      I1 = 2 * (J - 1)  + 1
      I2 = I1 + 1
      DERIV(1) = X(I1)
      DERIV(2) = 0.
      DERIV(3) = X(I2)
      DERIV(4) = 0.
      DERIV(5) = 0.
      DERIV(6) = X(I1)
      DERIV(7) = 0.
      DERIV(8) = X(I2)
      DERIV(9) = 1.0D0
      DERIV(10) = 0.
      DERIV(11) = 0.
      DERIV(12) = 1.0D0
      YC(1) = GP1(1) * X(I1)  +  GP1(2) * X(I2)  +  GP1(5)
      YC(2) = GP1(3) * X(I1)  +  GP1(4) * X(I2)  +  GP1(6)
      RETURN
C
      END
```

```
C
      SUBROUTINE BIORTH (A, B, N)
C
C     WHERE-  (1) A = AN N-BY-N MATRIX
C             (2) B = THE LEFT INVERSE OF THE MATRIX A
C             (3) N = ROW- AND COLUMN-DIMENSION OF THE MATRICES
C
      DOUBLE PRECISION A(30,30), B(30,30), DOT, NORM, EPS, CJK,
     *                 NSA
      EXTERNAL DOT
C
      DATA EPS /1.D-9/
C
      COMMON /NORM/ NORM
C
C
      NIT = 0
      DO 100 I=1,N
      DO 100 J=1,N
      B(I,J) = A(I,J)
  100 CONTINUE
C
  200 CONTINUE
      NIT = NIT + 1
      DO 300 K=1,N
      CALL SCLR ((1.D0 / DOT (B(1,K), A(1,K), N)), B(1,K), B(1,K), N)
      DO 300 J=1,N
      IF (J .EQ. K) GO TO 300
      CJK = DOT (B(1,J), A(1,K), N)
      DO 301 I=1,N
      B(I,J) = B(I,J)  -  CJK * B(I,K)
C
  301 CONTINUE
  300 CONTINUE
      NORM = 0.0
      DO 400 I=1,N
      DO 400 J=1,N
      NORM = NORM + (DOT(B(1,I), A(1,J), N) - AMINO(I/J, J/I))**2
  400 CONTINUE
      NORM = DSQRT(NORM)
      IF (NORM .GT. ((2.**NIT) * EPS)) GO TO 200
C
      DO 500 I=1,N
      DO 500 J=1,N
      NSA = B(I,J)
      B(I,J) = B(J,I)
      B(J,I) = NSA
  500 CONTINUE
C
      RETURN
```

```
      SUBROUTINE SCLR (Z, X, Y, N)                                      SCL06300
      DOUBLE PRECISION X, Y, Z                                          SCL06400
      DIMENSION X(1), Y(1)                                              SCL06500
      DO 700 K=1,N                                                      SCL06600
      Y(K) = Z * X(K)                                                   SCL06700
  700 CONTINUE                                                          SCL06800
      RETURN                                                            SCL06900
      END                                                               SCL07000
```

```
      DOUBLE PRECISION FUNCTION DOT (X, Y, N)                            DOT05300
      DOUBLE PRECISION X, Y                                             DOT05400
      DIMENSION X(1), Y(1)                                              DOT05500
      DOT = 0.0                                                         DOT05600
      DO 600 K=1,N                                                      DOT05700
      DOT = DOT  +  X(K) * Y(K)                                         DOT05800
  600 CONTINUE                                                          DOT05900
      RETURN                                                            DOT06000
C                                                                       DOT06100
      END                                                               DOT06200
```

5-4. GEOMETRIC CORRECTION

1. NAME

GEOCOR

2. PURPOSE

Apply geometric corrections to large data sets. This program implements the affine transformation.

3. CALLING SEQUENCE

This is a main program. It is currently on a partitioned data set as an executable module.

4. INPUT-OUTPUT

4.1 Input

The following input parameters should be supplied on data cards according to the formats and read statements indicated below.

```
      READ 100,A,XOP,YOP
      READ 200, NREC,NEL,INTPL,INVFLG,SCALEX,SCALEY
100   FORMAT(6E12.3)
200   FORMAT(4I6,2F6.1)
```

where

A is a 2x2 matrix defining the geometric transformation.
XOP, YOP are the shifts defining the geometric transformation.
NREC=Number of records in the input image.
NEL=Number of pixels per record in the input image.
INTPL is a flag indicating the type of interpolation to be used in producing the output records. (0: nearest neighbor, 1: Bilinear, 2: Bicubic).
INVFLG is a flag indicating whether the given transformation should be inverted or should be used as supplied (1 for inversion).
SCALEX=Number of pixels per unit distance in the X-direction.
SCALEY=Number of pixels per unit distance in the Y-direction.
Note that SCALEX and SCALEY are floating point variables.

Table of Contents

Table of Contents (Cont'd.)

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (Cont'd.)

Figure

LIST OF TABLES

The input image data should be on Unit 8 with one record per scan line, NEL pixels per record and one word (4 bytes) per pixel.

4.2 Output

The output of this program will consist of a printout of the coordinates of the extremities of the output image, the input and output image sizes, the desired transformation matrix and shift vector, the implemented transformation matrix and shift vector (which in some cases, could be different; e.g. when a 70^{o} rotation is desired, a -20^{o} rotation will be performed instead and the result will have to be rotated by 90^{o} using a transposition program). Also, the inverse matrix and some implementation details are printed. The output image will appear on Unit 10 in the same format as input. Since the image after geometric transformation is not necessarily rectangular with the edges parallel and perpendicular to the scan lines, the "exterior" is filled with zeros and the output is stored as a rectangular image.

4.3 File Storage

This program requires space on two direct access units 90 and 91 depending on the core size supplied (MAXC, the dimension of the array IX), the input image size and the transformation desired. Currently 1500X1500 word direct access files are provided for. If these are not sufficient, the program prints an error message, specifies the number of records and words per record required for work areas of input (90) and output (91) images. The user should then change the DEFINE FILE statements and the values of NRW90, NRW91, MAXR90, MAXR91 in the source program, recompile and run the job. Here, NRW and MAXR refer to the number of records and number of words per record, respectively.

5. EXITS

No abnormal exits except as described in Section 4.3.

6. USAGE

The program is in FORTRAN IV and implemented on the IBM 360 using the H compiler. The program, in its executable form, is in the user's library.

7. EXTERNAL REFERENCES

As indicated below.

Calling Program	Programs Called
GEOCOR	ROTATM
ROTATM	ROTAT1 ROTAT2 ROTAT4 ROTAT3 ROTAT5 ROTAT6 ROTAT7
ROTAT1	VMOV
ROTAT4	ROTAT3
ROTAT5	READER RITER MOVVMR SVSCI
ROTAT6	DOT
ROTAT7	DAWN ROTAT3 DARN
READER	IRVCON RIVCON

8. PERFORMANCE SPECIFICATIONS

8.1 Storage

This program is 120840 bytes long (mainly due to array IX dimensioned 30000 words). Including external references and buffers it needs 184K bytes.

8.2 Execution Time

The time is largely dependent on the output image size (which is a function of the input image size and the transformation) and the type of interpolation used. Typical times for various output image sizes are shown below.

INTPL	OUTPUT IMAGE SIZE	EXECUTION TIME (MINUTES)
0	600X600	1
0	1600X2400	10

8.3 I/O Load

None except as specified by Section 4.

8.4 Restrictions

None.

9. METHOD

A detailed description of the method used for handling the geometric correction of large images is given elsewhere [21]. Here, we shall confine ourselves to the discussion of the scale factors SCALEX and SCALEY.

Suppose a matrix A and XOP, YOP are supplied to the routine ROTATM. Then the transformation applied is

$$\begin{bmatrix} XP \\ YP \end{bmatrix} = A \begin{bmatrix} X \\ Y \end{bmatrix} + \begin{bmatrix} XOP \\ YOP \end{bmatrix}$$

where XP, YP are the coordinates in the transformed system and X, Y are those in the original (input) system.

Now, if the output image should be enlarged by factors SCALEX and SCALEY in the X and Y directions, we should modify the above equations to

$$\begin{bmatrix} XP \\ YP \end{bmatrix} = \begin{bmatrix} SCALEX & 0 \\ 0 & SCALEY \end{bmatrix} \left\{ A \begin{bmatrix} X \\ Y \end{bmatrix} + \begin{matrix} XOP \\ YOP \end{matrix} \right\}.$$

As an example, consider the case where the matrix A and XOP, YOP supplied to this program are the six parameters determined by GEOGREF [22] . Then the transformation yields the Landsat pixel coordinates in terms of UTM coordinates. If the UTM coordinates are supplied in kilometers, the matrix A will be in Landsat pixels per kilometer. Now, if the data are to be corrected to UTM coordinates, we should supply INVFLG=1. Also, the resulting transformation will yield one pixel per kilometer. If it is desired to have, say 20 pixels per kilometer (a convenient scale close to Landsat resolution), we should supply SCALEX=SCALEY=20.0. Also, sometimes it is desirable to have line printer plots on which the physical scale in the X and Y direction are the same. But line printers generally print at 10 characters per inch and 6 lines per inch. Thus, to get the same number of kilometers per inch of plot we can choose (SCALEX,SCALEY)= (10,6) or (20,12) etc.

10. COMMENTS

It is possible that this program will exceed the estimated time. However, to get a better estimate of time needed for a subsequent run, the printed output provides an indication of how many output records have been processed. The "number of partitions" multiplied by the number of records in the output image is the total number of records to be processed. "Partitions" and "column groups" are used synonymously. A message is printed after processing every 500 records in each partition.

11. LISTINGS

This listing of the program, the associated subroutines are attached at the end.

12. TESTS

The program has been tested thoroughly for several sizes of images and various geometric transformations. A sample output is attached.

```
C     MAIN PROGRAM TO APPLY GEOMETRIC CORRECTION.          GEOCOR
C
      DATA MAXC/30000/
      DIMENSION IX(30000),A(2,2)
      LOGICAL*1 LX(4,1500),LY(1500)
      EQUIVALENCE(IX(1),LX(1)),(IX(1501),LY(1))
      DEFINE FILE 90(1500,1500,U,IAV90)
      DEFINE FILE 91(1500,1500,U,IAV91)
      COMMON/AFINE/C(2,2),B(2,2),XPP,YPP,INTPL,IA(2,2),ILO,IHI,JLO,JHI
      COMMON/WRKDSK/NRW90,MAXR90,NRW91,MAXR91
      NRW90=1500
      NRW91=1500
      MAXR90=1500
      MAXR91=1500
C
C     READ TRANSFORMATION TO BE USED
C     MATRIX A AND XOP,YOP DEFINE THE TRANSFORMATION
C
      READ 100,A,XOP,YOP
100   FORMAT(6E12.3)
C
C     READ IN IMAGE SIZE AND THE TYPE OF INTERPOLATION TO BE USED
C
C     INTPL=0,1,2 FOR NEAREST NEIGHBOR, BILINEAR AND BICUBIC
      READ 200,NREC,NEL,INTPL,INVFLG,SCALEX,SCALEY
200   FORMAT(4I6,2F6.1)
      IF(INVFLG.EQ.1)GO TO 40
      A(1,1)=A(1,1)*SCALEX
      A(1,2)=A(1,2)*SCALEX
      A(2,1)=A(2,1)*SCALEY
      A(2,2)=A(2,2)*SCALEY
      XOP=XOP*SCALEX
      YOP=YOP*SCALEY
      GO TO 50
40    CONTINUE
C
C     INVERT THE TRANSFORMATION.
C
      DET=A(1,1)*A(2,2)-A(1,2)*A(2,1)
      C(1,1)=A(2,2)/DET
      C(2,1)=-A(2,1)/DET
      C(1,2)=-A(1,2)/DET
      C(2,2)=A(1,1)/DET
      A(1,1)=C(1,1)*SCALEX
      A(1,2)=C(1,2)*SCALEX
      A(2,1)=C(2,1)*SCALEY
      A(2,2)=C(2,2)*SCALEY
      XPP=A(1,1)*XOP+A(1,2)*YOP
      YPP=A(2,1)*XOP+A(2,2)*YOP
      XOP=-XPP
      YOP=-YPP
50    CONTINUE
C
C     PERFORM CORRECTION.
C
      CALL ROTATM(IX,MAXC,NREC,NEL,A,XOP,YOP,8,10,NRECO,NELO)
      STOP
      END
```

```
      SUBROUTINE ROTATM(IX,MAXC,NREC,NEL,AMAT,XPO,YPO,NTAPI,NTAPO,
     .       NRECO,NELO)
C
      DIMENSION IX(MAXC), AMAT(2,2)
      COMMON/AFINE/A(2,2),B(2,2),XOP,YOP,INTPL,IA(2,2),ILO,IHI,JLO,JHI
      COMMON/ROT56/IRINIT
      COMMON/WRKDSK/NRW90,MAXR90,NRW91,MAXR91
C
C     DEFINE FILE 90(NRW90,MAXR90*4,L,IAV90)
C     DEFINE FILE 91(NRW91,MAXR91*4,L,IAV91)
C
C     COMPUTE OUTPUT IMAGE ARRAY SIZE.
C
      WRITE (6,1000)
      CALL ROTAT1(NREC,NEL,AMAT,XPO,YPO,NRECO,NELO)
      WRITE(6,1010)NREC,NEL
      WRITE(6,1100)NRECO,NELO
      WRITE(6,1200)((AMAT(I,J),J=1,2),I=1,2),XPO,YPO
      WRITE(6,1210)((A(I,J),J=1,2),I=1,2),XOP,YOP
      WRITE(6,1220)((B(I,J),J=1,2),I=1,2)
 1000 FORMAT (1H1)
 1010 FORMAT(22H  INPUT PICTURE SIZE=(,I5,1H,,I5,1H))
 1100 FORMAT(22H OUTPUT PICTURE SIZE=(,I5,1H,,I5,1H))
 1200 FORMAT(///30H DESIRED TRANSFORMATION MATRIX,2(/2E15.4),//,
     .21H DESIRED SHIFT VECTOR,2(/E15.4))
 1210 FORMAT(///34H IMPLEMENTED TRANSFORMATION MATRIX,2(/2E15.4),//,
     .25H IMPLEMENTED SHIFT VECTOR,2(/E15.4))
 1220 FORMAT(///15H INVERSE MATRIX,2(/2E15.4))
C
C      COMPUTE DIMENSIONS OF INPUT ARRAY WHICH CAN BE HELD IN CORE AT A
C      TIME AND NUMBER OF PARTITIONS REQUIRED.
C
      MAXCP=MAXC-MAX0(NEL,NELO)
      MAXCP1=MAXCP+1
      CALL ROTAT2(MAXCP,NEL,NR,NC,NCGP)
      WRITE(6,1230)MAXC,MAXCP
      IF(INTPL.EQ.0) WRITE(6,1240)
      IF(INTPL.EQ.1) WRITE(6,1241)
      IF(INTPL.EQ.2) WRITE(6,1242)
      WRITE(6,1300)NR,NC
      WRITE(6,1400)NCGP
 1230 FORMAT(20H MAX. CORE SUPPLIED=,I6,/,
     .37H MAX. CORE AVAILABLE FOR INPUT ARRAY=,I6)
 1240 FORMAT(//38H RESAMPLING METHOD--- NEAREST  NEIGHBOR)
 1241 FORMAT(//44H RESAMPLING METHOD--- BILINEAR  INTERPOLATION)
 1242 FORMAT(//44H RESAMPLING METHOD---  BICUBIC  INTERPOLATION)
 1300 FORMAT(27H TEMP. 2D CORE ARRAY SIZE=(,I5,1H,,I5,1H))
 1400 FORMAT(19H NO. OF PARTITIONS=,I3)
C
C     FIND NUMBER OF WORK RECORDS AND LONGEST RECORD LENGTH TO BE
C     WRITTEN ON AUXILIARY DISK MODULE FOR INTERMEDIATE OUTPUT.
```

```
C
      CALL ROTAT4(NCGP,NC,NRECO,NELO,MAXR,NROUT)
C
C     FIND AND PRINT SIZE OF DIRECT ACCESS WORKFILES REQUIRED FOR INPUT
C     OUTPUT.
C
      NELDI=NEL-IFIX(B(2,2)+FLOAT(NC-4))
      IF(NCGP.LE.1)GO TO 50
      PRINT 1500,  NREC,NELDI,NROUT,MAXR
      IF(NREC.LE.NRW90.AND.NELDI.LE.MAXR9).
           AND.NROUT.LE.NRW91.AND.MAXR.LE.MAXR91)GO TO 50
      NRECO=0
      NELO=0
      PRINT 1600
1500  FORMAT(' NO. OF RECORDS IN INPUT WORK FILE='I5/
           ' NO. OF WORDS PER RECORD IN INPUT WORK FILE='I5/
            NO. OF RECORDS IN OUTPUT WORK FILE='I5/
           ' NO. OF WORDS PER RECORD IN OUTPUT WORK FILE='I5)
1600  FORMAT(//' ERROR CONDITION IN ROTATN.  SUPPLIED WORKFILE SPACE IS
      INSUFFICIENT.  THEREFORE THE PROGRAM RETURNED WITH NRECO=NELO=0')
      RETURN
C
50    CONTINUE
      DO 10 ICGP=1,NCGP
      I20=0
      I21=1
      YLOC=(ICGP-1)*IFIX(B(2,2)+FLOAT(NC-4))+1
      IRINIT=1
      NC1=NC
      IF(ICGP.EQ.NCGP)NC1=NEL-(NCGP-1)*IFIX(B(2,2)+FLOAT(NC-4))
      DO 20 IREC=1,NRECO
      CALL ROTAT3 (ICGP,NCGP,NC,IREC,NELO,JP1,JP2,I2,I)
C
C     UPDATE ARRAY IX IF NECESSARY. IX IS TREATED AS IF IT WERE
C     DIMENSIONED (NR,NC)
C     I20=LAST INPUT ROW NUMBER PRESENTLY IN CORE. I2=LAST INPUT ROW
C     NUMBER NEEDED FOR COMPUTING IREC'TH ROW OF OUTPUT.
C     I21=ROW NUMBER THE INPUT TAPE IS READY TO READ.
C
      CALL ROTAT5(I20,I21,I2,IX,IX(MAXCP1),NREC,NEL,NR,NC,ICGP,NCGP,
     .     NTAPI)
C
C     GENERATE IY(JP1) THRU IY(JP2) USING IX.
C
      CALL ROTAT6(IX,IX,IX(MAXCP1),NR,NC1,IREC,JP1,JP2,YLOC,NREC,NEL)
C
C     WRITE IY(JP1) THRU IY(JP2) ON DISC OR TAPE DEPENDING ON ICGP VALU
C
      CALL ROTAT7 (ICGP,NCGP,IREC,NELO,NC,JP1,JP2,IX(MAXCP1),NTAPO)
      IF(MOD(IREC,500).EQ.0)PRINT 1700,  IREC,ICGP
20    CONTINUE
10    CONTINUE
      RETURN
1700  FORMAT(' FINISHED PROCESSING'I5,' RECORDS IN COLUMN GROUP'I3)
      END
```

```
      SUBROUTINE ROTATI(NREC,NEL,AMAT,XPO,YPO,NRECO,NELO)
C
C     GIVEN MATRIX A AND XOP, YOP FIND B=INVERSE(A). DECIDE IF A SHD
C     BE MODIFIED FOR CONVENIENCE OF IMPLEMENTATION AND FIND THE TRUE
C     INITIAL AND FINAL COORDINATES OF THE OUTPUT IMAGE GENERATED.
C     TRANSFORMATION TO BE APPLIED IS
C         XP=A*X+XO, WHERE A IS A 2X2 MATRIX AND XP IS A 2-VECTOR.
C
      COMMON/AFINE/A(2,2),B(2,2),XOP,YOP,INTPL,IA(2,2),ILO,IHI,JLO,JHI
      DIMENSION AMAT(2,2)
      CALL VMOV(AMAT,4,A)
      XOP=XPO
      YOP=YPO
      DET=A(1,1)*A(2,2)-A(1,2)*A(2,1)
      B(1,1)=A(2,2)/DET
      B(2,2)=A(1,1)/DET
      B(1,2)=-A(1,2)/DET
      B(2,1)=-A(2,1)/DET
      IA(1,1)=1
      IA(1,2)=0
      IA(2,1)=0
      IA(2,2)=1
      P=NREC
      Q=NEL
C
C     IF DET.LE.1.E-8 PRINT MESSAGE.
C
      IF(DET .LE.1.E-8)WRITE(6,100)
      IF(DET .LE.1.E-8)PRINT 100
100   FORMAT(1X,51HCAUTION*** REQUESTED TRANSFORMATION MAY BE SINGULAR)
      RAT1=ABS(B(1,1)/B(2,1))
      RAT2=ABS(B(1,2)/B(2,2))
      IF(ABS(B(2,1)).LE.1.E-8)RAT1=1.E20
      IF(ABS(B(2,2)).LE.1.E-8)RAT2=1.E20
      IFLG=0
      IF(RAT1.GE.RAT2) GO TO 10
      IFLG=1
      IA(1,1)=0
      IA(1,2)=1
      IA(2,1)=1
      IA(2,2)=0
      W=B(1,1)
      B(1,1)=B(1,2)
      B(1,2)=W
      W=B(2,1)
      B(2,1)=B(2,2)
      B(2,2)=W
10    IF(B(1,1).GE.0.)GO TO 20
      IFLG=1
      IA(1,1)=-IA(1,1)
      IA(2,1)=-IA(2,1)
```

```
      B(1,1)=-B(1,1)
      B(2,1)=-B(2,1)
20    CONTINUE
      IF(B(2,2).GE.0.)GO TO 30
      IFLG=1
      IA(1,2)=-IA(1,2)
      IA(2,2)=-IA(2,2)
      B(1,2)=-B(1,2)
      B(2,2)=-B(2,2)
30    CONTINUE
      IPASS=1
      WRITE(6,151)
151   FORMAT(/)
      WRITE(6,150)
150    FORMAT(/,72H COORDINATES OF IMAGE EXTREMITIES IF DESIRED TRANSFO
     .MATION IS PERFORMED)
C
C     COMPUTE COORDINATES OF TOP, BOTTOM, LEFT, RIGHT CORNERS OF THE
C     IMAGE IN THE TRANSFORMED COORDINATE SYSTEM.
C
50    CONTINUE
      W1=A(1,2)+XOP
      W2=W1+A(1,1)*P
      W1=A(1,1)+W1
      W3=A(1,2)*Q+XOP
      W4=W3+A(1,1)*P
      W3=A(1,1)+W3
      RLO=AMIN1(W1,W2,W3,W4)
      ILO=RLO
      IF(RLO.GT.0..AND.RLO.NE.FLOAT(ILO))ILO=ILO+1
      RHI=AMAX1(W1,W2,W3,W4)
      IHI=RHI
      IF(RHI.LT.0..AND.RHI.NE.FLOAT(IHI))IHI=IHI-1
      WRITE(6,200)RLO,ILO,RHI,IHI
200   FORMAT(/,8H TOP ROW,F10.2,3H OR,I6,11H BOTTOM ROW,F10.2,3H OR,I6)
      W1=A(2,2)+YOP
      W2=W1+A(2,1)*P
      W1=A(2,1)+W1
      W3=A(2,2)*Q+YOP
      W4=W3+A(2,1)*P
      W3=A(2,1)+W3
      RLO=AMIN1(W1,W2,W3,W4)
      JLO=RLO
      IF(RLO.GT.0..AND.RLO.NE.FLOAT(JLO))JLO=JLO+1
      RHI=AMAX1(W1,W2,W3,W4)
      JHI=RHI
      IF(RHI.LT.0..AND.RHI.NE.FLOAT(JHI))JHI=JHI-1
      WRITE(6,300)RLO,JLO,RHI,JHI
300   FORMAT(/,12H LEFT COLUMN,F10.2,3H OR,I6,13H RIGHT COLUMN,F10.2,3H
     OR,I6)
      IF(IFLG.EQ.0.OR.IPASS.EQ.2)GO TO 40
      IPASS=2
      WRITE(6,151)
      WRITE(6,350)
350   FORMAT(/,54H COORDINATES OF EXTREMITIES OF IMAGE ACTUALLY PRODUCE
```

```
      )
      IDET=IA(1,1)*IA(2,2)-IA(1,2)*IA(2,1)
      IW=IA(2,2)/IDET
      IA(2,2)=IA(1,1)/IDET
      IA(1,1)=IW
      IA(1,2)=-IA(1,2)/IDET
      IA(2,1)=-IA(2,1)/IDET
      W1=FLOAT(IA(1,1))*A(1,1)+FLOAT(IA(1,2))*A(2,1)
      W2=FLOAT(IA(1,1))*A(1,2)+FLOAT(IA(1,2))*A(2,2)
      W3=FLOAT(IA(2,1))*A(1,1)+FLOAT(IA(2,2))*A(2,1)
      W4=FLOAT(IA(2,1))*A(1,2)+FLOAT(IA(2,2))*A(2,2)
      A(1,1)=W1
      A(1,2)=W2
      A(2,1)=W3
      A(2,2)=W4
      W1=FLOAT(IA(1,1))*XOP+FLOAT(IA(1,2))*YOP
      W2=FLOAT(IA(2,1))*XOP+FLOAT(IA(2,2))*YOP
      XOP=W1
      YOP=W2
      GO TO 50
   40 CONTINUE
      NRECO=IHI-ILO+1
      NELO=JHI-JLO+1
      RETURN
      END
```

```
      SUBROUTINE ROTAT2(MAXC,NEL,NR,NC,NCGP)
C
C     FIND MAX NO. OF RECORDS THAT MUST BE HELD IN CORE TO GENERATE
C     LONGEST POSSIBLE OUTPUT RECORD GIVEN CORE SIZE LIMIT.  ALSO, FIND
C     NO. OF COLUMN GROUPS INTO WHICH INPUT DATA SHD BE SECTIONED.
C
      COMMON/AFINE/A(2,2),B(2,2),XOP,YOP,INTPL,IA(2,2),ILO,IHI,JLO,JHI
      TT=ABS(B(1,2)/B(2,2))
      J=MAXC/3
      IF(TT.LE.1.E-10)GO TO 20
      C=MAXC
      TT2=2.*TT
      NC1=(TT-5.+SQRT((TT-5.)**2+4.*TT*(C+TT2-5.)))/TT2
      NC2=(TT-4.+SQRT((TT-4.)**2+4.*TT*(C+TT2-4.)))/TT2+1.
      DO 10 I=NC1,NC2
      J=NC1+NC2-I
      II=TT*FLOAT(J-2)
      IF((II+5)*(J+1).LE.MAXC)GO TO 20
   10 CONTINUE
   20 NC=MINO(NEL,J+1)
      NR=MAXC/NC

C     FIND NCGP, THE NUMBER OF COLUMN GROUPS.

      NCGP=1
      NCP=IFIX(B(2,2))+NC-4
      IF(NC.LT.NEL)NCGP=(NEL-NC)/NCP+2
      RETURN
      END
```

```
      SUBROUTINE ROTAT3 (ICGP,NCGP,NC,IREC,NELO,JP1,JP2,I2,IFLG)
C
C     FOR GIVEN OUTPUT RECORD NUMBER IREC AND PARTITION NUMBER ICGP, TO
C     FIND OUTPUT WORD NUMBERS THAT CAN BE COMPUTED AND INPUT RECORD
C     NUMBERS REQUIRED.
C
      COMMON/AFINE/A(2,2),B(2,2),XOP,YOP,INTPL,IA(2,2),ILO,IHI,JLO,JHI
      Y1=1.+FLOAT((ICGP-1)*(NC-4+IFIX(B(2,2))))
      Y2=Y1+FLOAT(NC-1)
      XP=IREC+ILO-1
      W=YOP-B(2,1)*(XP-XOP)/B(2,2)
      YP1=(Y1+1.)/B(2,2)+W
      YP2=(Y2-1.)/B(2,2)+W
      JP1=YP1
      IF(YP1.GT.0..AND.YP1.NE.FLOAT(JP1))JP1=JP1+1
      JP2=YP2
      IF(YP2.LT.0..OR.YP2.EQ.FLOAT(JP2))JP2=JP2-1
      KP1=MIN0(NELO,MAX0(1,JP1-JLO+1))
      KP2=MIN0(NELO,MAX0(1,JP2-JLO+1))
      JP1=KP1
      JP2=KP2
      IF(ICGP.EQ.1)JP1=1
      IF(ICGP.EQ.NCGP)JP2=NELO
      IF(IFLG.EQ.0)RETURN
C
      YP2=KP2+JLO-1
      IF(B(1,2).LT.0.)YP2=KP1+JLO-1
      X2=B(1,1)*(XP-XOP)+B(1,2)*(YP2-YOP)
      I2=X2+2.
      RETURN
      END
```

```
      SUBROUTINE ROTAT4(NCGP,NC,NRECD,NELD,MAXR,NROUT)
      IF(NCGP.EQ.1)RETURN
      MAXR=0
      CALL ROTAT3 (1,NCGP,NC,1,NELD,JP1,JP2,I2,0)
      MAXR=MAX0(MAXR,JP2-JP1)
      CALL ROTAT3 (1,NCGP,NC,NRECD,NELD,JP1,JP2,I2,0)
      MAXR=MAX0(MAXR,JP2-JP1)
      CALL ROTAT3 (NCGP,NCGP,NC,NRECD,NELD,JP1,JP2,I2,0)
      MAXR=MAX0(MAXR,JP2-JP1)
      CALL ROTAT3 (NCGP,NCGP,NC,1,NELD,JP1,JP2,I2,0)
      MAXR=MAX0(MAXR,JP2-JP1)
      MAXR=MAXR+1
      NROUT=NRECD*(NCGP-1)
      RETURN
      END
```

```
      SUBROUTINE ROTATS(I20,I21,I2,IX,IY,NREC,NEL,NR,NC,ICGP,NCGP,NTAPI)
      DIMENSION IX(NR,NC),IY(NEL)
      COMMON/ROTSE/IRINIT
      COMMON/AFINE/A(2,2),B(2,2),XCP,YCP,INTPL,IA(2,2),ILO,IHI,JLO,JHI
C
C     FIND NUMBER OF RECORDS TO BE READ.
C
C
      IF(I21.GT.NREC)RETURN
      NROWS=I2-I20
      IF(NROWS.LE.0)RETURN
C
C     IF ICGP=1 READ DATA FROM INPUT TAPE. ELSE READ FROM DISK.
C     IF NCGP.NE.1 AND ICGP=1 WRITE THE LAST NEL-NC+4 WORDS OF INPUT ON
C     DISK UNIT 90.
C
      IF(ICGP.EQ.1)IDEV=NTAPI
      IF(ICGP.NE.1)IDEV=0
      JDEV=0
      IF(ICGP.NE.1.OR.NCGP.EQ.1)JDEV=-1
      NC1=IFIX(B(2,2)*FLOAT(NC-3))
      NEL2=NEL-NC1+1
      NEL1=1
      IF(ICGP.NE.1)NEL1=(ICGP-2)*(NC1-1)+1
      ITYPO=0
      IF(INTPL.NE.0)ITYPO=1
      DO 10 I=1,NROWS
      J=I
      IF(ICGP.EQ.1)CALL READER(IDEV,IY,IY,NEL,I21,0,ITYPO)
      IF(ICGP.GT.1)CALL READER(IDEV,IY,IY,NEL2,I21,1,1)
      CALL WRITER(JDEV,IY(NC1),I21,NEL2)
      I21=I21+1
      I20=I20+1
C
C     MOVE THE APPROPRIATE PART OF IY INTO MOD(IRINIT+I-2,NR)+1'TH ROW
C     OF IX.
C
      CALL MOVVMK(IX,NR,NC,IY(NEL1),MOD(IRINIT+I-2,NR)+1)
      IF(I21.LE.NREC)GO TO 10
      IDEV=-1
      JDEV=-1
      CALL SVSCI(IY,NEL,0)
      GO TO 20
10    CONTINUE
20    IRINIT=MOD(IRINIT+J-1,NR)+1
C
C     NEW IRINIT IS THE ROW NUMBER IN IX CONTAINING THE EARLIEST RECORD
C     OF THE INPUT IMAGE.
C
      RETURN
      END
```

```
      SUBROUTINE ROTAT6 (IX,RX,IY,NR,NC,IREC,JP1,JP2,YLOC,NREC,NEL)
      DIMENSION IX(NR,NC), RX(NR,NC), IY(1)
      COMMON/AFINE/AM(2,2),BM(2,2),XOP,YOP,INTPL,IA(2,2),ILO,IHI,JLO,J
      IRND(A)=MAX0(0,MIN0(63,IFIX(A+.5)))
C
C     GENERATE IY(JP1) THRU IY(JP2) USING IX.
C     INTPL=0, 1, 2 RESULT IN NEAREST, BILINEAR, BICUBIC INTERPOLATION
C     RESAMPLING
C
      IF(INTPL.GT.0)GO TO 11
      P=NREC
      Q=NEL
      YLOC1=YLOC-1.5
      XP=IREC+ILO-1
      XP=XP-XOP
      YOP1=FLOAT(JLO-1)-YOP
      DO 10 JP=JP1,JP2
      YP=FLOAT(JP)+YOP1
      X=BM(1,1)*XP+BM(1,2)*YP
      IF(X.LT.1..OR.X.GT.P)GO TO 20
      Y=BM(2,1)*XP+BM(2,2)*YP
      IF(Y.LT.1..OR.Y.GT.Q)GO TO 20
      I=X+.5
      I=MOD(I-1,NR)+1
      J=Y-YLOC1
      IY(JP)=IX(I,J)
      GO TO 10
20    IY(JP)=0
10    CONTINUE
      RETURN
C
11    IF(INTPL.GT.1)GO TO 12
      P=NREC
      Q=NEL
      YLOC1=YLOC-1.
      XP=IREC+ILO-1
      XP=XP-XOP
      YOP1=FLOAT(JLO-1)-YOP
      DO 101 JP=JP1,JP2
      YP=FLOAT(JP)+YOP1
      X=BM(1,1)*XP+BM(1,2)*YP
      IF(X.LT.1..OR.X.GT.P)GO TO 201
      Y=BM(2,1)*XP+BM(2,2)*YP
      IF(Y.LT.1..OR.Y.GT.Q)GO TO 201
      B=Y-YLOC1
      I=X
      J=B
      A=X-FLOAT(I)
      A1=1.-A
      B=B-FLOAT(J)
      I=MOD(I-1,NR)+1
```

```
      I1=MOD(I,NR)+1
      J1=J+1
      IY(JP)=(A1*RX(I,J)+A*RX(I1,J))*(1.-B)+(A1*RX(I,J1)+A*RX(I1,J1))*B
     .5
      GO TO 101
201   IY(JP)=0
101   CONTINUE
      RETURN
C
12    CONTINUE
      DIMENSION H(4),W(4)

      P=NREC-1
      P1=NREC
      Q=NEL-1
      Q1=NEL
      YLOC1=YLOC-1.
      XP=IREC+ILO-1
      XP=XP-XOP
      YOP1=FLOAT(JLO-1)-YOP
      DO 102 JP=JP1,JP2
      YP=FLOAT(JP)+YOP1
      X=BM(1,1)*XP+BM(1,2)*YP
      IF(X.LT.2..OR.X.GT.P)GO TO 202
      Y=BM(2,1)*XP+BM(2,2)*YP
      IF(Y.LT.2..OR.Y.GT.Q)GO TO 502
      I=X
      B=Y-YLOC1
      J=B
      A=X-FLOAT(I)
      A1=1.-A
      B=B-FLOAT(J)
      B1=1.-B
      AA1=A*A1
      AA11=AA1+1.
      H(1)=-A1*AA1
      H(2)=A1*AA11
      H(3)=A*AA11
      H(4)=-A*AA1
      I=MOD(I-2,NR)+1
      I1=MOD(I,NR)+1
      I2=MOD(I1,NR)+1
      I3=MOD(I2,NR)+1
      J=J-2
      DO 302 JJ=1,4
      JK=J+JJ
302   W(JJ)=H(1)*RX(I,JK)+H(2)*RX(I1,JK)+H(3)*RX(I2,JK)+H(4)*RX(I3,JK)
      BB1=B*B1
      BB11=BB1+1.
      H(1)=-B1*BB1
      H(2)=B1*BB11
      H(3)=B*BB11
      H(4)=-B*BB1
      IY(JP)=IRND(DOT(W,H,4))
      GO TO 102
```

```
202   IF(X.LT.1..OR.X.GT.P1)GO TO 402
      Y=BM(2,1)*XP+BM(2,2)*YP
502   IF(Y.LT.1..OR.Y.GT.Q1)GO TO 402
      B=Y-YLOC1
      I=X
      J=B
      H(2)=X-FLOAT(I)
      H(1)=1.-H(2)
      I=MOD(I-1 ,NR)+1
      I1=MOD(I,NR)+1
      J1=J+1
      W(1)=H(1)*RX(I,J)+H(2)*RX(I1,J)
      W(2)=H(1)*RX(I,J1)+H(2)*RX(I,J1)
      H(2)=B-FLOAT(J)
      H(1)=1.-H(2)
      IY(JP)=DOT(W,H,2)+.5
      GO TO 102
402   IY(JP)=-1
102   CONTINUE
      RETURN
      END
```

```
      SUBROUTINE ROTAT7 (ICGP,NCGP,IREC,NELO,NC,JP1,JP2,IY,NTAPO)
      DIMENSION IY(NELO)
C
C     IF NCGP=1, WRITE IY ON NTAPO
C     IF ICGP.NE.NCGP WRITE IY ON DISK UNIT 91. IF ICGP=NCGP READ PARTS
C     OF IY CORRESPONDING TO JCGP.LT.NCGP FROM DISK AND WRITE ASSEMBLED
C     IY ON NTAPO.
C
      IF (NCGP.EQ.1) GO TO 10
C
      NCGP1=NCGP-1
      NDAREC=NCGP1*(IREC-1)
      IF (ICGP.EQ.NCGP) GO TO 20
C
      CALL DAWN (91,NDAREC+ICGP,IY(JP1),4*(JP2-JP1+1))
      RETURN
C
20    DO 30 JCGP=1,NCGP1
      CALL ROTAT8 (JCGP,NCGP,NC,IREC,NELO,KP1,KP2,0,0)
      CALL DARN (91,NDAREC+JCGP,IY(KP1),4*(KP2-KP1+1))
30    CONTINUE
C
10    WRITE(NTAPO)IY
      RETURN
      END
```

```
      SUBROUTINE VMOV(IX,N,IY)
C
C     TO MOVE VECTOR IX INTO VECTOR IY.
C
      DIMENSION IX(N),IY(N)
      IF(N.EQ.0)RETURN
      DO 10 I=1,N
10    IY(I)=IX(I)
      RETURN
      END
```

```
      SUBROUTINE READER(IDEV,IX,X,NEL,IREC,ITYPI,ITYPO)
      DIMENSION IX(NEL),X(NEL)
C
C     THIS ROUTINE RETURNS IF IDEV.LT.0.  IT READS NEXT RECORD ON SEQUEN
C     TIAL UNIT IF IDEV.GE.2.  IT READS IREC'TH RECORD ON DIRECT ACCESS
C     UNIT IF IDEV.LT.2.AND IDEV.GE.0.
C     TYPE CONVERSION IS AN OPTION. ITYPI AND ITYPO REPRESENT INPUT AND
C     OUTPUT DATA TYPES RESPECTIVELY. (0-- INTEGER. 1-- REAL).
C
C     MUST EQ'CE(IX,X)
C
      IF(IDEV.LT.0)RETURN
      IF(IDEV.LT.2)GO TO 20
      READ(IDEV)X
      GO TO 30
20    IDEV90=IDEV+90
      READ(IDEV90'IREC)X
30    IF(ITYPI.EQ.ITYPO)RETURN
      IF(ITYPI.EQ.0.AND.ITYPO.EQ.1)CALL IRVCON(IX,X,NEL)
      IF(ITYPI.EQ.1.AND.ITYPO.EQ.0)CALL RIVCON(IX,X,NEL)
      RETURN
      END
```

```
      SUBROUTINE IRVCON(IX,X,N)
      DIMENSION IX(N),X(N)
      DO 10 I=1,N
10    X(I)=IX(I)
      RETURN
      ENTRY RIVCON(IX,X,N)
      DO 20 I=1,N
20    IX(I)=X(I)
      RETURN
      END
```

```
      SUBROUTINE RITER(IDEV,X,IREC,NEL)
C
C     TO WRITE NEL WORDS OF X ON DEVICE IDEV. IF IDEV.LT.0 RETURN.
C     IF(IDEV.GT.1) (IX(IEL),IEL=1,NEL) IS WRITTEN AS ONE RECORD ON TAP(
C     IF IDEV IS 0 OR 1 THE RECORD IS WRITTEN ON IDEV+90, A DISC UNIT.
C
      DIMENSION X(NEL)
      IF(IDEV.LT.0)RETURN
      IF(IDEV.LT.2)GO TO 20
      WRITE(IDEV)X
      RETURN
20    IDEV90=IDEV+90
      WRITE(IDEV90'IREC)X
      RETURN
      END
```

```
      SUBROUTINE MOVVMR(A,M,N,X,I)
C
C     MOVE N VECTOR X TO I'TH ROW OF M*N MATRIX A.
C
      DIMENSION A(M,N),X(N)
      DO 10 J=1,N
10    A(I,J)=X(J)
      RETURN
C
      ENTRY MOVMRV(A,M,N,X,I)
C
C     MOVE I'TH ROW OF A TO X.
C
      DO 20 J=1,N
20    X(J)=A(I,J)
      RETURN
      END
```

```
      SUBROUTINE SVSCI(IX,N,IS)
      DIMENSION IX(N)
      DO 10 I=1,N
10    IX(I)=IS
      RETURN
      END
```

```
      FUNCTION DOT(X,Y,N)
      DIMENSION X(N),Y(N)
      DOT=0.
      DO 10 I=1,N
   10 DOT=DOT+X(I)*Y(I)
      RETURN
      END
```

```
      SUBROUTINE DARN(IDEV,IREC,X,N)
      LOGICAL*1 X(N)
      READ(IDEV'IREC)X
      RETURN
      ENTRY DAWN(IDEV,IREC,X,N)
      WRITE(IDEV'IREC)X
      RETURN
      ENTRY SARN(NTAPI,X,N)
      READ(NTAPI)X
      RETURN
      ENTRY SAWN(NTAPO,X,N)
      WRITE(NTAPO)X
      RETURN
      END
```

```
COORDINATES OF IMAGE EXTREMITIES IF DESIRED TRANSFORMATION IS PERFORMED

TOP ROW        2.11 OR      3 BOTTOM ROW   1350.07 OR  1350

LEFT COLUMN    -271.44 OR  -271 RIGHT COLUMN   1015.10 OR  1015
 INPUT PICTURE SIZE=(   625,   755)
OUTPUT PICTURE SIZE=(  1348,  1287)


DESIRED TRANSFORMATION MATRIX
     0.1859E 01      0.2504E 00
    -0.4364E 00      0.1345E 01

DESIRED SHIFT VECTOR
     0.0
     0.0


IMPLEMENTED TRANSFORMATION MATRIX
     0.1858E 01      0.2504E 00
    -0.4364E 00      0.1345E 01

IMPLEMENTED SHIFT VECTOR
     0.0
     0.0


INVERSE MATRIX
     0.5158E 00     -0.9603E-01
     0.1674E 00      0.7123E 00
MAX. CORE SUPPLIED= 50000
MAX. CORE AVAILABLE FOR INPUT ARRAY= 48713


RESAMPLING METHOD--- NEAREST NEIGHBOR
TEMP. 2D CORE ARRAY SIZE=(    83,   586)
NO. OF PARTITIONS=  2
NO. OF RECORDS IN INPUT WORK FILE=   625
NO. OF WORDS PER RECORD IN INPUT WORK FILE=  173
NO. OF RECORDS IN OUTPUT WORK FILE= 1348
NO. OF WORDS PER RECORD IN OUTPUT WORK FILE= 1092
FINISHED PROCESSING  500 RECORDS IN COLUMN GROUP  1
FINISHED PROCESSING 1000 RECORDS IN COLUMN GROUP  1
FINISHED PROCESSING  500 RECORDS IN COLUMN GROUP  2
FINISHED PROCESSING 1000 RECORDS IN COLUMN GROUP  2

 ELAPSED TIME   --   MINUTES -  3  SECONDS - 29.20
```

GEOMETRIC TRANSFORMATION OF CURVES - I

1 NAME: GETC1

2 PURPOSE: To apply a given geometric transformation to a curve given in SLIC (scan line intersection code) format. This routine generates the row and column coordinates of all points on the curve in the transformed coordinate system and writes them on a direct access device. It can handle cases where the number of row and column coordinates produced exceeds the core capacity.

3 CALLING SEQUENCE:

CALL GETC1(NTAPI,IY1,MAXC,IDUM,IY2,A,XPO,YPO,NTOT,IRC,
IWR,IWC,MINR,MAXR,NREC)

where

IY1, IY2, IRC, IWR, IWC are work arrays to be dimensioned as indicated in the attached listings.

NTAPI = Logical unit number of the input sequential data set (input).

MAXC = Core capacity available for row (or column) coordinates produced (input, should be as large as possible).

IDUM = Number of dumps on to the direct access device (output).

A = Matrix defining the geometric transformation (rotation, skew and scale change) (input).

(XPO, YPO) = Vector defining the geometric transformation (translation) (input).

NTOT = Number of row (or column) coordinates in the final dump on the direct access device (output).

MINR, MAXR = Vectors containing the minima and maxima of the row coordinates in each of the dumps in the output (output).

NREC = Number of records in the input data (input).

4 INPUT-OUTPUT:

The input, consisting of NREC records, is in the SLIC format on a sequential file (e.g., output of SMOB)

The output of this routine consists of 2*IDUM records on a direct access file (logical unit 90). Every $(2*I-1)^{st}$ record consists of a set of row coordinates arranged in ascending order. Every $(2*I)^{th}$ record consists of the corresponding set of column coordinates. The first 2*(IDUM-1) records have MAXC words each and each of the last two records has NTOT words.

5 EXITS: No nonstandard exits.

6 USAGE: The program is in FORTRAN IV and is implemented on the IBM 360/65 system. An IBM 7094 version is also available.

7 EXTERNAL INTERFACES:

7.1 System Subroutines: IBCOM#

7.2 Other Routines Called: SORT, JOIN1, VMOV.

7.3 External Storage: None

8 PERFORMANCE SPECIFICATIONS:

8.1 Storage: 8B2 Hexadecimal locations.

8.2 Execution Time: Depends on the image size, MAXC and the transformation to be implemented. The timing for a test case for both GETC1 and the next step, GETC2, is shown.

8.3 I/O Load: None

8.4 Restrictions: None

9 METHOD:

Initially NTOT = 1 and IDUM = 0. For each record of input which has a nonzero number of boundary points, the following computations are performed. The column coordinates in the record are sorted in ascending order (this is not necessary if the input records contain data already in ascending order). The column coordinates IY(J) in the I^{th} record are examined one by one. If $IY(J+1) - IY(J) \leq 1$, then the routine JOIN1 is called to generate a straight line in the transformed coordinate system between (I, IY(J)) and (I, IY(J+1)). Also, if there are points in the $(I+1)^{st}$ record connected to the point (I, IY(J)), then the routine JOIN1 is used to generate straight lines joining them to it in the transformed system.

While handling any record, the current record is held in IY1 and the next record is held in IY2. After finishing each record, IY2 is moved to IY1 and a new record is read into IY2.

The routine JOIN1 works as follows. Given two points (X1, Y1) and (X2, Y2), the transformed coordinates (XP1, YP1) and (XP2, YP2) are computed using

$$\begin{bmatrix} XPi \\ YPi \end{bmatrix} = A \begin{bmatrix} Xi \\ Yi \end{bmatrix} + \begin{bmatrix} XP0 \\ YP0 \end{bmatrix}$$

Next, a digital approximation to the straight line joining (XP1, YP1) to (XP2, YP2) is found using a routine "JOIN". The row and column coordinates of the points on this line are stored in arrays IWR and IWC. The number of such points after one call to JOIN is given by K. The total number of points computed and held in the array IRC is given by NTOT-1. Now, if there are any points (IWR(I), IWC(I)) which are identical to points in IRC, then they are eliminated and K is corrected accordingly. If $NTOT+K \leq MAXC+1$, then IWR, IWC are moved into

IRC and NTOT is set to NTOT+K. Otherwise, the parts of IWR, IWC corresponding to the first MAXC+1 - NTOT points are moved into IRC and the array IRC is dumped on the direct access device 90 as two records of length MAXC each. Now, the remainder of IWR, IWC is moved into IRC and NTOT is changed to K - (MAXC - NTOT). Also IDUM is set to IDUM+1.

After all the records have been processed, NTOT is changed to (NTOT-1), the data in IRC are dumped on Unit 90 as two records of length NTOT each and IDUM is set to IDUM+1.

10 COMMENTS: None

11 LISTINGS: The listings for GETC1 follow along with GETC2 and the subroutines required.

GEOMETRIC TRANSFORMATION OF CURVES - II

1 NAME: GETC2

2 PURPOSE: To rearrange the row and column coordinates on the direct access file (produced by a routine such as GETC1) in SLIC format and write on a sequential file.

3 CALLING SEQUENCE:

```
CALL GETC2(IDUM,IA,IDA,ISEQ,ISKIP,MAXC,MINR,MAXR,
           NWDS,IRC,NTAPO,IRMN,IRMX,ICMN,ICMX)
```

where

IDUM = Number of sets of row and column coordinates to be read from the direct access file (Unit 90).
= 1/2 (Number of records on the file) (input)

IA is an array dimensioned (IDA,5) with MINR,MAXR,ISEQ, NWDS and ISKIP equivalenced to IA(1,1), ..., IA(1,5), respectively.

IDA is a number greater than or equal to IDUM. (input)

ISEQ, ISKIP and IRC are work arrays.

MAXC = Maximum core capacity available for reading the row and column data. IRC is dimensioned (MAXC,2). (input)

MINR,MAXR = Arrays containing the minima and maxima of row coordinates in each of the "row records" on the input file (input).

NWDS = Array containing the number of words to be read from each of the "row (or column) records" on the input file (input).

NTAPO = Logical unit number of the output sequential data set (input).

IRMN, IRMX = Minimum and maximum row coordinates for the entire image (output).

ICMN, ICMX = Minimum and maximum column coordinates for the entire image (output).

4 INPUT-OUTPUT

The input data for this routine is on a direct access file (Unit 90) which has 2*IDUM records. The input value of NWDS(I) indicates the number of words of relevant data in the (2*I-1)st and the (2*I)th records. Each odd numbered record contains the row coordinates followed, in the next record, by the corresponding column coordinates. The row coordinates must be in ascending order.

The output consists of IRMX-IRMN+1 records written in SLIC format on unit NTAPO, the first record corresponding to the IRMN'th row of the image.

5 EXITS: No nonstandard exits. However, there is an error exit in the case where the supplied MAXC is not sufficient to handle the data. In this case, an error message is printed and IRMX is set to IRMN-1.

6 USAGE: The program is in FORTRAN IV and is implemented on IBM 360/65. An IBM 7094 version is also available.

7 EXTERNAL INTERFACES:

7.1 System Subroutines: IBCOM#

7.2 Other Routines Called: VMINI4, VMAXI4, SORT, VMOV

7.3 External Storage: None

8 PERFORMANCE SPECIFICATIONS:

8.1 Storage: C3C Hexadecimal bytes

8.2 Execution Time: Depends on the size and content of the input file. A test for the geometric correction of TARCOG county boundary data consisting of 1553 records with approximately 12000 boundary points took about four minutes with MAXC=7995.

8.3 I/O Load: None

8.4 Restrictions: None

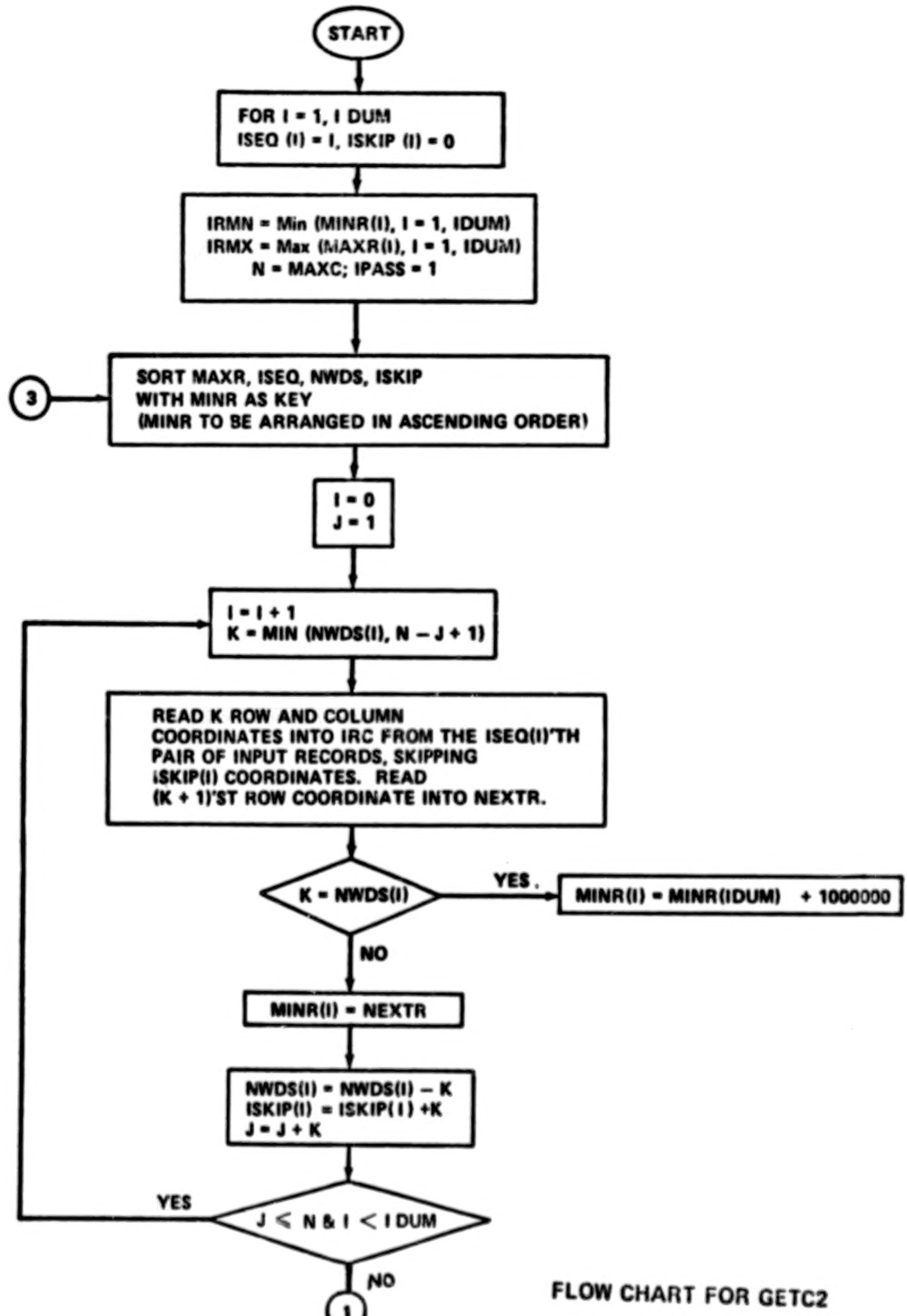
9 METHOD:

A flow chart describing the steps in the routine is shown. Briefly, the algorithm consists in (i) reading as much of the data as possible into core, (ii) sorting them in ascending order of row coordinates, (iii) finding the largest row coordinate $\bar{R}$ in core that has no unread parts on the input file, (iv) reformatting and writing the column coordinates corresponding to the data in core whose row coordinates are less than or equal to $\bar{R}$ and (v) repeating steps (i) through (iv) until all the data are processed.

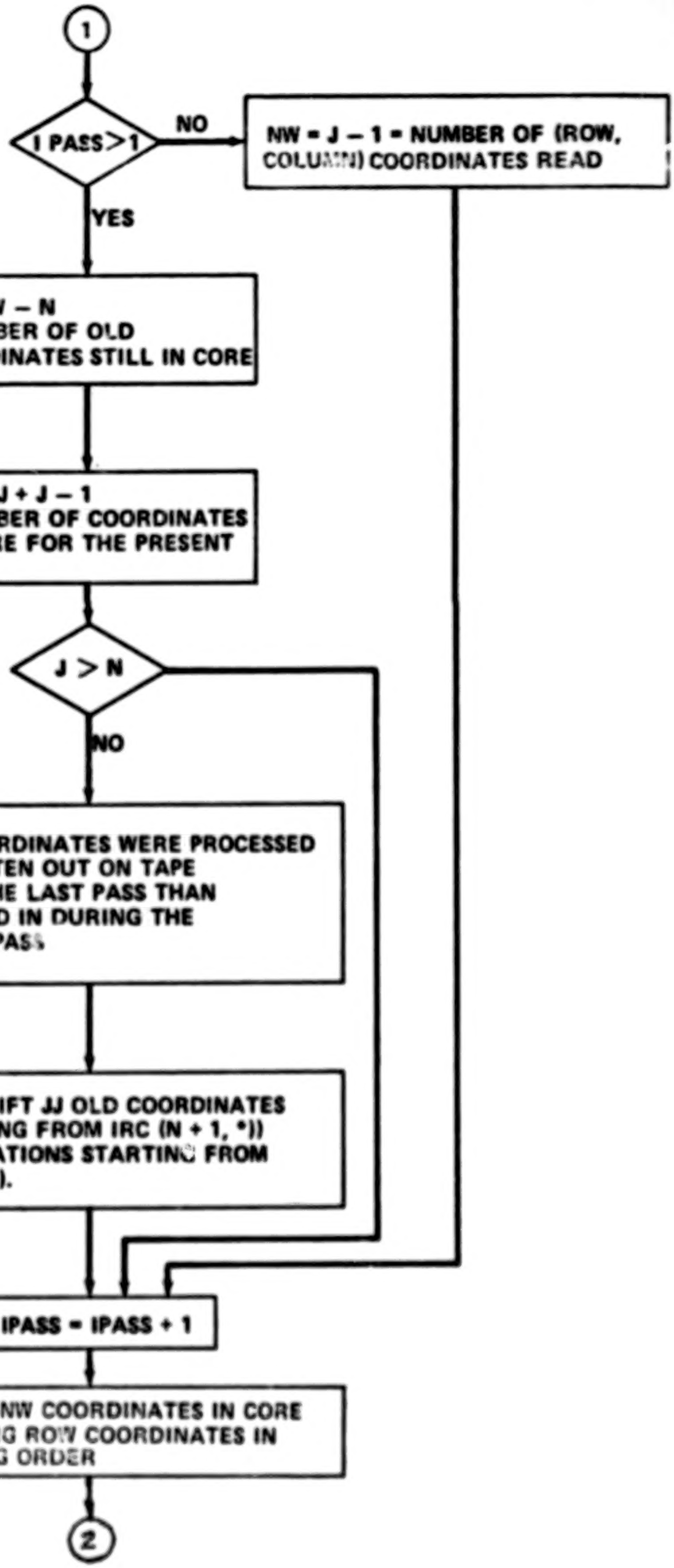
10 COMMENTS: None

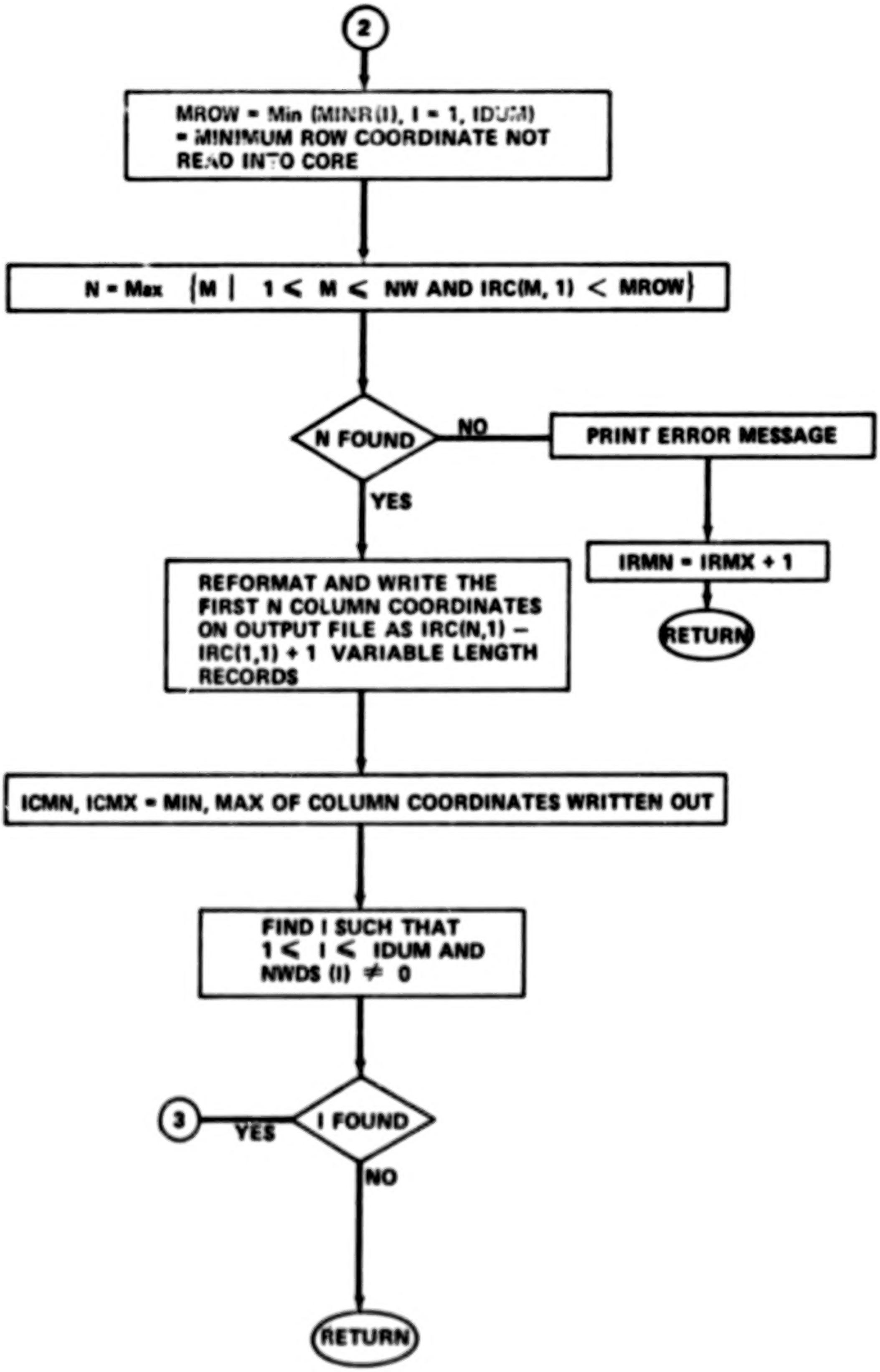
11 LISTINGS: The listings for GETC1, GETC2 and the subroutines required are attached at the end of this section.

12 TESTS: The routines GETC1 and GETC2, in combination, have been tested using test patterns and on the TARCOG county boundary map. Visual inspection of the picture of the corrected TARCOG county boundaries superposed on a land use classification scanner data indicates accurate performance of the programs.



FLOW CHART FOR GETC2


1
I PASS > 1
NO
NW = J − 1 = NUMBER OF (ROW, COLUMN) COORDINATES READ
YES
JJ = NW − N
= NUMBER OF OLD COORDINATES STILL IN CORE
NW = JJ + J − 1
= NUMBER OF COORDINATES IN CORE FOR THE PRESENT PASS
J > N
NO
MORE COORDINATES WERE PROCESSED AND WRITTEN OUT ON TAPE DURING THE LAST PASS THAN WERE READ IN DURING THE CURRENT PASS
LEFT-SHIFT JJ OLD COORDINATES (STARTING FROM IRC (N + 1, *)) TO LOCATIONS STARTING FROM IRC (J, *).
IPASS = IPASS + 1
SORT THE NW COORDINATES IN CORE ARRANGING ROW COORDINATES IN ASCENDING ORDER
2


2
MROW = Min (MINR(I), I = 1, IDUM)
= MINIMUM ROW COORDINATE NOT
READ INTO CORE
N = Max {M | 1 ≤ M ≤ NW AND IRC(M, 1) < MROW}
N FOUND
NO
PRINT ERROR MESSAGE
IRMN = IRMX + 1
RETURN
YES
REFORMAT AND WRITE THE
FIRST N COLUMN COORDINATES
ON OUTPUT FILE AS IRC(N,1) –
IRC(1,1) + 1 VARIABLE LENGTH
RECORDS
ICMN, ICMX = MIN, MAX OF COLUMN COORDINATES WRITTEN OUT
FIND I SUCH THAT
1 ≤ I ≤ IDUM AND
NWDS (I) ≠ 0
I FOUND
YES
3
NO
RETURN

```
          [illegible] IZ(200)
          DIMENSION IA(20,5),ISEQ(20),ISKIP(20),NWDS(20)
          DIMENSION IY1(125),IY2(125),A(2,2),IRC(500,2),INR(20),INC(20),
     .        MINR(20),MAXR(20)
          EQUIVALENCE((IA(1)),MINR(1)),(IA(21)),MAXR(1)),(IA(41),ISEQ(1)),
     .        (IA(61)),NWDS(1)),(IA(81)),ISKIP(1)))
          DEFINE FILE 90(20,500,U,IAV90)
          DATA NTAPI,MAXC,NREC/10,500,60/
          DATA A,XPO,YPO/.707,-.707,.707,.707,0.,0./
          DATA NTAPW/12/
          DATA IDA,NTAPO/20,8/
          CALL GETC1(NTAPI,IY1,MAXC,IDUM,IY2,A,XPO,YPO,NTOT,IRC,
     .        INR,INC,MINR,MAXR,NREC)
          CALL SVSC1(NWDS,IDUM,MAXC)
          NWDS(IDUM)=NTOT
          CALL GETC2(IDUM,IA,IDA,ISEQ,ISKIP,MAXC,MINR,MAXR,NWDS,IRC,
     .        NTAPO,IRMN,IRMX,ICMN,ICMX)
          REWIND NTAPO
          PRINT 100,  IRMN,IRMX,ICMN,ICMX
  100     FORMAT(' MIN. ROW NO.='I5/
     .        ' MAX. ROW NO.='I5/
     .        ' MIN. COL. NO.='I5/
     .        ' MAX. COL. NO.='I5)
          NRECO=IRMX-IRMN+1
          IF(NRECO.EQ.0)STOP
          NELO=ICMX-ICMN+1
          DO 10 I=1,NRECO
          READ(NTAPO) N,(IY1(J),J=1,N)
          CALL SETB(IY1,N,IZ,NELO,ICMN,[illegible])
   10     WRITE(NTAPW)(IZ(J),J=1,NELO)
          REWIND NTAPW
          NPAGE=(NELO-1)/125+1
          DO 20 IP=1,NPAGE
          J1=(IP-1)*125+1
          IF(IP.EQ.NPAGE.AND.NELO.GT.124)J1=NELO-124
          J2=MINO(J1+124,NELO)
          K1=J1+ICMN-1
          K2=J2+ICMN-1
          PRINT 200,  K1,K2
  200     FORMAT('1GEOMETRICALLY CORRECTED BOUNDARY IMAGE-  COLUMNS'I5,
     .        ' THROUGH'I5,'.')
          DO 30 I=1,NRECO
          READ(NTAPW)(IZ(J),J=1,NELO)
   30     PRINT 300 ,  (IZ(J),J=J1,J2)
  300     FORMAT(1X125A1)
   20     REWIND NTAPW
          STOP
          END
```

```
      ISN 0002          SUBROUTINE GETC1(NTAPI,IY1,MAXC,IDUM,IY2,A,XPO,YPO,NTOT,IAC,
                       1 IWR,IWC,MINR,MAXR,NREC)                                    00003890
      ISN 0003          COMMON/JON1/I                                               00003900
      ISN 0004          COMMON/JON2/KMAX                                            00003910
                  C
                  C     TO APPLY A GEOMETRIC TRANSFORMATION TO A GIVEN SET OF CURVES    00003940
                  C     WHOSE COORDINATE DATA ARE STORED ON NTAPI AND GENERATE NEW COORDINO0003950
                  C     DATA.  THE OUTPUT WILL BE ON DISK . CONNECTIVITY WILL BE PRESERVED00003960
                  C     ISOLATED POINTS WILL BE SUPPRESSED.                         00003970
                  C
                  C     DEFINE FILE 9012=((EXPECTED NO. OF BOUNDARY POINTS)/MAXC+1,MAXC,
                  C     U,IAV90)
                  C     D'N IY1 AND IY2 MAX. NO. OF WORDS EXPECTED IN ANY RECORD OF INPUT.
                  C     D'N IWR AND IWC AS IN JOINER.
                  C     D'N MINR AND MAXR NUMBER OF EXPECTED SUBDIVISIONS OF DATA NEEDED T
                  C     TO COMPUTE ALL BOUNDARY DATA. I.E., EXPECTED VALUE OF IDUM)
      ISN 0005          DIMENSION IY1(1),IY2(1), A(2,2),IAC(MAXC,2),IWR(1),IWC(1)
      ISN 0006          DIMENSION T(2),TT(2)
      ISN 0007          DIMENSION MINR(1),MAXR(1)                                   00004000
      ISN 0008          READ(NTAPI)NC1,(IY1(J),J=1,NC1)                             00004010
      ISN 0009          MAXC1=MAXC+1
      ISN 0010          CALL SORT(IY1,1,NC1,NC1,1,T,TT)
      ISN 0011          NTOT=1                                                      00004030
      ISN 0012          IWREC=1
      ISN 0013          IDUM=0                                                      00004090
      ISN 0014          KMAX=0                                                      00004100
      ISN 0015          DO 10 I=1,NREC                                              00004110
      ISN 0016          IF(I.LT.NREC)READ(NTAPI)NC2,(IY2(J),J=1,NC2)                00004120
      ISN 0018          CALL SORT(IY2,1,NC2,NC2,1,T,TT)
                  C
                  C     IF(NC1.EQ.0)NO COMPUTATIONS ARE NEEDED. GO TO END OF LOOP AND UP- 00004150
                  C     DATE THE ARRAYS.                                            00004160
                  C
      ISN 0019          IF(NC1.EQ.0)GO TO 20                                        00004180
                  C
                  C     COMPUTE LINES CORRESPONDING TO CONNECTED POINTS IN THE ARRAY IY1(J00004200
                  C
      ISN 0021          X1=I                                                        00004220
      ISN 0022          JJ1=1                                                       00004230
      ISN 0023          DO 30 J1=1,NC1                                              00004240
      ISN 0024          IF(J1.EQ.NC1)GO TO 40                                       00004250
      ISN 0026          J2=J1+1                                                     00004260
      ISN 0027          IF(IY1(J2)-IY1(J1).GT.1)GO TO 40                            00004270
      ISN 0029          X2=X1                                                       00004280
      ISN 0030          Y1=IY1(J1)                                                  00004290
      ISN 0031          Y2=IY1(J2)                                                  00004300
      ISN 0032          CALL JOIN1(X1,Y1,X2,Y2,A,XPO,YPO,MAXC1,MAXC,NTOT,IWREC,IDUM,IAC,
                       1 IWR,IWC,MINR,MAXR)
      ISN 0033     40   IF(NC2.EQ.0.OR.I.EQ.NREC)GO TO 30                           00004330
                  C
                  C     COMPUTE LINES CORRESPONDING TO CONNECTIONS BETWEEN (I-1)'ST AND 00004350
                  C     I'TH ROWS.                                                  00004360
                  C
      ISN 0035          X2=I+1                                                      00004380
      ISN 0036          Y1=IY1(J1)                                                  00004390
      ISN 0037          DO 60 JJ=JJ1,NC2                                            00004400
      ISN 0038          JJ2=JJ                                                      00004410
      ISN 0039          IDIF=IY2(JJ)-IY1(J1)                                        00004420
      ISN 0040          IF(IDIF.LT.-1)GO TO 60                                      00004430
      ISN 0042          IF(IDIF.GT.1)GO TO 70                                       00004440
      ISN 0044          Y2=IY2(JJ)                                                  00004450
      ISN 0045          CALL JOIN1(X1,Y1,X2,Y2,A,XPO,YPO,MAXC1,MAXC,NTOT,IWREC,IDUM,IAC,
                       1 IWR,IWC,MINR,MAXR)
      ISN 0046     60   CONTINUE                                                    00004480
      ISN 0047     70   JJ1=MAX0(1,JJ2-1)                                           00004490
      ISN 0048     30   CONTINUE                                                    00004500
      ISN 0049          IF(I.EQ.NREC)GO TO 10                                       00004510
```

```
                C
                C      UPDATE ARRAYS IY1 AND IY2                                     00004530
                C
  ISN 0051      20     NC1=NC2                                                       00004540
  ISN 0052             CALL VMOV(IY2,NC2,IY1)                                        00004550
  ISN 0053      10     CONTINUE                                                      00004560
  ISN 0054             NTOT=NTOT-1                                                   00004570
  ISN 0055             IF(NTOT.EQ.0)RETURN                                           00004580
  ISN 0057             CALL SORT(IRC,1,NTOT,MAXC,2,T,TT)
  ISN 0058             WRITE(90'IWREC)(IRC(IEL,1),IEL=1,NTOT)
  ISN 0059             IWREC=IWREC+1
  ISN 0060             WRITE(90'IWREC)(IRC(IEL,2),IEL=1,NTOT)
  ISN 0061             IWREC=IWREC+1
  ISN 0062             IDUM=IDUM+1                                                   00004620
  ISN 0063             MINR(IDUM)=IRC(1,1)
  ISN 0064             MAXR(IDUM)=IRC(NTOT,1)
  ISN 0065             PRINT 1000,  KMAX                                             00004650
  ISN 0066             WRITE(6,1000)KMAX                                             00004660
  ISN 0067    1000     FORMAT(6H KMAX=I6)                                            00004670
  ISN 0068             RETURN                                                        00004680
  ISN 0069             END                                                           00004690


  ISN 0002             SUBROUTINE GETC2(IDUM,IA,IDA,ISEQ,ISKIP,MAXC,MINR,MAXR,NWDS,IRC,
                      .     NTAPO,IRMN,IRMX,ICMN,ICMX)
  ISN 0003             DIMENSION IA(IDA ,5),T(5),TT(5)
  ISN 0004             DIMENSION ISEQ(IDA),ISKIP(IDA),MINR(IDA),MAXR(IDA),NWDS(IDA),
                      .     IRC(MAXC,2)
                C
                C      DEFINE FILE 90 AS IN GETC1.
                C      MUST EQ'CE CLMS 1,2,3,4,5 OF IA TO MINR,MAXR,ISEQ,NWDS,ISKIP.
                C
                C      INITIALIZE ISEQ, ISKIP AND N.
                C
  ISN 0005             DO 10 I=1,IDUM
  ISN 0006             ISEQ(I)=1
  ISN 0007      10     ISKIP(I)=0
  ISN 0008             N=MAXC
  ISN 0009             IPASS=1
  ISN 0010             IRMN=10**6
  ISN 0011             ICMN=10**6
  ISN 0012             IRMX=-IRMN
  ISN 0013             ICMX=-ICMN
  ISN 0014             CALL VMINI4(MINR,IDUM,IRMN)
  ISN 0015             CALL VMAXI4(MAXR,IDUM,IRMX)
                C
                C      ARRANGE MINR IN ASCENDING ORDER AND MAXR,ISEQ,NWDS,ISKIP ACCLY.
                C
  ISN 0016             NREC=0
  ISN 0017      20     CALL SORT(IA,1,IDUM,IDA ,5,T,TT)
  ISN 0018             PRINT 100,  IPASS,NREC,(MINR(I),MAXR(I),ISEQ(I),NWDS(I),ISKIP(I),
                      .     I=1,IDUM)
  ISN 0019     100     FORMAT(//' PASS'I3,'.  NUMBER OF RECORDS DUMPED OUT='I5/
                      .   (' MINR='I6,' MAXR='I6,' ISEQ='I3,' NWDS='I6,' ISKIP='I6))
                C
                C      READ N WORDS INTO ARRAYS IROW AND ICLM FROM DISK WITH
                C      ISEQ(1)'ST SET OF DATA.  WORDS ISKIP(J)+1 THRU ISKIP(J)+NWDS(J)
                C      ARE READ FROM ISEQ(J)'TH SET OF DATA.
                C
  ISN 0020             I=0
  ISN 0021             J=1
  ISN 0022      30     I=I+1
  ISN 0023             IWREC=(ISEQ(I)-1)*2+1
  ISN 0024             ISKIPI=ISKIP(I)
  ISN 0025             K=MINO(NWDS(I),N-J+1)
  ISN 0026             JK1=J+K-1
  ISN 0027             IF(ISKIPI.EQ.0)
                      .READ(90'IWREC)                       ((IRC(IEL,1),IEL=J,JK1),NEXTR
  ISN 0029             IF(ISKIPI.NE.0)
                      .READ(90'IWREC)(DUM,IEL=1,ISKIPI),((IRC(IEL,1),IEL=J,JK1),NEXTR
  ISN 0031             IF(K.LT.NWDS(I))MINR(I)=NEXTR
  ISN 0033             IF(K.EQ.NWDS(I))MINR(I)=MINR(IDUM)+1000000
  ISN 0035             IF(ISKIPI.EQ.0)
                      .READ(90'IWREC+1)                         ((IRC(IEL,2),IEL=J,JK1)
  ISN 0037             IF(ISKIPI.NE.0)
```

```
      READ(90'IWREC+1)(DUM,IEL=1,ISKIP1),(IRC(IEL,2),IEL=J,JK1)
      NWDS(I)=NWDS(I)-K
      ISKIP(I)=ISKIP(I)+K
      J=J+K
      IF(J.LE.N.AND.I.LT.IDUM)GO TO 30
C
C     (J-1) WDS HAVE NOW BEEN READ FROM DISK INTO CORE
C     NW=NUMBER OF WDS IN CORE ON THE PREVIOUS PASS OUT OF WHICH N HAVE
C     BEEN TRANSFERRED TO OUTPUT-TAPE.  THUS (NW-N) IS THE NO. OF OLD
C     WDS STILL IN CORE.
C
      IF(IPASS.GT.1)GO TO 40
      NW=J-1
      GO TO 45
   40 JJ=NW-N
      NW=JJ+J-1
C
C     NW=NUMBER OF WORDS IN CORE FOR THE PRESENT PASS
C
      IF(J.GT.N)GO TO 45
      N1=N+1
      CALL VMOV(IRC(N1,1),JJ,IRC(J,1))
      CALL VMOV(IRC(N1,2),JJ,IRC(J,2))
   45 CONTINUE
      IPASS=IPASS+1
C
C     SORT ROW AND COLUMN
C
      CALL SORT(IRC,1,NW,MAXC,2,T,TT)
C
C     FIND PART OF DATA IN CORE TO BE REFORMATTED AND WRITTEN ON TAPE.
C     THAT IS, FIND MAX, N SUCH THAT IROW(N).LT.MROW WHERE MROW=MINIMUM
C     ROW NUMBER CORRESPONFING TO DATA ON DISK THAT HAVE NOT BEEN TAKEN
C     INTO CORE YET.
C
      MROW=MINR(1)
      CALL VMINI4(MINR,IDUM,MROW)
      NW1=NW+1
      DO 60 J=1,NW
      N=NW1-J
      IF(IRC(N,1).LT.MROW)GO TO 50
   60 CONTINUE
C
C     NO PART OF DATA IN CORE CAN BE WRITTEN ON TAPE BECAUSE THE RECORDS
C     ARE INCOMPLETE.  PRINT ERROR MESSAGES AND STOP.
C
      PRINT 400,NREC
      IRMN=IRMX+1
      RETURN
  400 FORMAT(' ERROR IN GETC2.  TRY LARGER MAXC.  NREC='I6)
   50 CONTINUE
C
C     WRITE N WORDS OF COLUMN DATA ON TAPE AFTER REFORMATTING.
C
      J=0
      J1=1
   70 J=J+1
      IF(J.EQ.N)GO TO 80
      IF(IRC(J,1).EQ.IRC(J+1,1))GO TO 70
      JJ=J-J1+1
      NREC=NREC+1
      WRITE(NTAPO)JJ,(IRC(L,2),L=J1,J)
      CALL VMINI4(IRC(J1,2),JJ,ICMN)
      CALL VMAXI4(IRC(J1,2),JJ,ICMX)
      J1=J+1
      IF(IRC(J,1).EQ.IRC(J1,1)-1)GO TO 70
```

```
                   I1=IRC(J,1)+1
                   I2=IRC(J1,1)
                   JJ=0
                   DO 90 II=I1,I2
                   NREC=NREC+1
        90         WRITE(NTAPO)JJ,JJ
                   GO TO 70
        80         JJ=N-J1+1
                   NREC=NREC+1
                   WRITE(NTAPO)JJ,(IRC(L,2),L=J1,N)
                   CALL VMINI4(IRC(J1,2),JJ,ICMN)
                   CALL VMAXI4(IRC(J1,2),JJ,ICMX)
        C
        C          CHECK STOPPING CONDITION.  IF NOT FINISHED GO TO 20
        C
                   PRINT 200,  NREC
       200         FORMAT(//' FINISHED GETC2.- NO. OF RECORDS='I6)
                   DO 95 II=1,IDUM
                   IF(NWDS(II).NE.0)GO TO 20
        95         CONTINUE
                   RETURN
                   END


                  SUBROUTINE JOIN1(X1,Y1,X2,Y2,A,XPO,YPO,MAXC1,MAXC,NTOT,IWREC,IDUM,
                 .      IRC,IWR,IWC,MINR,MAXR)
                  DIMENSION A(2,2),IRC(MAXC,2),IWR(1),IWC(1),MINR(1),MAXR(1),T(2),
                 .      TT(2)
                  COMMON/JON1/I
                  COMMON/JON2/KMAX
        C
        C         D'N IWR,IWC AS IN JOIN. D'N MINR,MAXR((EXPECTED NUMBER OF POINTS
        C         IN TRANSFORMED DATA-1)/MAXC+1).
        C
                  CALL JOIN(X1,Y1,X2,Y2,A,XPO,YPO,IWR,IWC,K)
                  M=MIN0(K*2,NTOT-1)
                  IF(M.EQ.0)GOTO 10
                  MM=NTOT-M
                  CALL ELRPTN(IWR,IWC,IRC(MM,1),IRC(MM,2),K,M)
       10         CONTINUE
                  KMAX=MAX0(K,KMAX)
                  IF(K.EQ.0)RETURN
                  IF(NTOT+K.GT.MAXC1)GO TO 50
                  CALL VMOV(IWR,K,IRC(NTOT,1))
                  CALL VMOV(IWC,K,IRC(NTOT,2))
                  NTOT=NTOT+K
                  RETURN
       50         L=MAXC1-NTOT
                  CALL VMOV(IWR,L,IRC(NTOT,1))
                  CALL VMOV(IWC,L,IRC(NTOT,2))
                  CALL SORT(IRC,1,MAXC,MAXC,2,T,TT)
                  WRITE(90'IWREC)(IRC(IEL,1),IEL=1,MAXC)
                  IWREC=IWREC+1
                  WRITE(90'IWREC)(IRC(IEL,2),IEL=1,MAXC)
                  IWREC=IWREC+1
                  IDUM=IDUM+1
                  MINR(IDUM)=IRC(1,1)
                  MAXR(IDUM)=IRC(MAXC,1)
                  L1=L+1
                  K=K-L
                  CALL VMOV(IWR(L1),K,IRC(1,1))
                  CALL VMOV(IWC(L1),K,IRC(1,2))
                  NTOT=K+1
      200         FORMAT(6H KMAX=I6)
                  WRITE(6,200)KMAX
                  WRITE(6,100)I,IDUM,MINR(IDUM),MAXR(IDUM)
      100         FORMAT(/,1X,34HNUMBER OF INPUT RECORDS PROCESSED=I6/
                 .      1X,16HNUMBER OF DUMPS=I6,6H MINR=I6,6H MAXR=I6)
                  RETURN
                  END
```

```
      SUBROUTINE JOIN(X1,Y1,X2,Y2,A,XPO,YPO,IWR,IWC,K)
C
C     TO FIND INTEGER COORDINATES OF POINTS NEAREST TO THE LINE JOINING
C     (X1,Y1) AND (X2,Y2) IN THE TRANSFORMED SYSTEM WITH COORDINATE
C     TRANSFORMATION GIVEN BY A AND (XPO,YPO)
C
C     IWR AND IWC SHD BE D'NED MAX EXPECTED VALUE OF K. K=NUMBER OF
C     POINTS ON THE LINE BETWEEN GIVEN POINTS.
C
      IRND(B)=SIGN(ABS(B)+.5,B)
C
      DIMENSION A(2,2),IWR(1),IWC(1)
      XP1=A(1,1)*X1+A(1,2)*Y1+XPO
      XP2=A(1,1)*X2+A(1,2)*Y2+XPO
      YP1=A(2,1)*X1+A(2,2)*Y1+YPO
      YP2=A(2,1)*X2+A(2,2)*Y2+YPO
      W=AMIN1(XP1,XP2)
      I1=W
      IF(W.LT.FLOAT(I1))I1=I1-1
      W=AMAX1(XP1,XP2)
      I2=W
      IF(W.GT.FLOAT(I2))I2=I2+1
      I21=I2-I1+1
      W=AMIN1(YP1,YP2)
      J1=W
      IF(W.LT.FLOAT(J1))J1=J1-1
      W=AMAX1(YP1,YP2)
      J2=W
      IF(W.GT.FLOAT(J2))J2=J2+1
      J21=J2-J1+1
      K=0
      XP12=ABS(XP1-XP2)
      YP12=ABS(YP1-YP2)
      IF(XP12.LT.YP12)GO TO 10
      SLOPE=(YP2-YP1)/(XP2-XP1)
      I11=I1-1
      DO 20 I=1,I21
      K=K+1
      IWR(K)=I+I11
      XP=IWR(K)
   20 IWC(K)=IRND(YP1+(XP-XP1)*SLOPE)
      L=K
      DO 40 I=1,L
      IF(I.EQ.1)GO TO 40
      I1=I-1
      IIS=IWC(I)-IWC(I1)
      II=IABS(IIS)
      IF(II.LE.1)GO TO 40
      IS=IIS/II
      DO 50 J=2,II
      K=K+1
      IWR(K)=IWR(I1)
   50 IWC(K)=IWC(I1)+(J-1)*IS
   40 CONTINUE
      RETURN
   10 J11=J1-1
      SLOPE=(XP2-XP1)/(YP2-YP1)
      DO 30 J=1,J21
      K=K+1
      IWC(K)=J+J11
      YP=IWC(K)
   30 IWR(K)=IRND(XP1+(YP-YP1)*SLOPE)
      L=K
      DO 60 I=1,L
      IF(I.EQ.1)GO TO 60
      I1=I-1
      IIS=IWR(I)-IWR(I1)
      II=IABS(IIS)
      IF(II.LE.1)GO TO 60
      IS=IIS/II
      DO 70 J=2,II
      K=K+1
      IWC(K)=IWC(I1)
   70 IWR(K)=IWR(I1)+(J-1)*IS
   60 CONTINUE
      RETURN
      END
```

```
ISN 0002          SUBROUTINE ELAPTN(IWR,IWC,IROW,ICLM,K,N)
ISN 0003          DIMENSION IWR(1),IWC(1),IROW(N),ICLM(N)
             C
             C    TO ELIMINATE COORDINATES IN (IWR,IWC) WHICH ARE EQUAL TO ANY OF
             C    THE COORDINATES IN (IROW,ICLM).
             C    D'N IWR(K),IWC(K).  K IS BOTH INPUT AND OUTPUT (NUMBER OF COORDI-
             C    NATES).
             C
ISN 0004          I=0
ISN 0005     30   I=I+1
ISN 0006          IF(I.GT.K)RETURN
ISN 0008          DO 10 J=1,N
ISN 0009          IF(IWR(I).NE.IROW(J).OR.IWC(I).NE.ICLM(J))GO TO 10
ISN 0011          GO TO 20
ISN 0012     10   CONTINUE
ISN 0013          GO TO 30
ISN 0014     20   IF(I.EQ.K)GO TO 40
ISN 0016          CALL VMOV(IWR(I+1),K-I,IWR(I))
ISN 0017          CALL VMOV(IWC(I+1),K-I,IWC(I))
ISN 0018          I=I-1
ISN 0019          K=K-1
ISN 0020          GO TO 30
ISN 0021     40   K=K-1
ISN 0022          RETURN
ISN 0023          END
```

5-5 SUPERPOSITION OF BOUNDARIES

5-5-1 THINNING OF BOUNDARY IMAGES

1 NAME

PEELS

2 PURPOSE

Starting with the output of a **microdensitometer** digitizing a boundary image, to apply a given threshold of density and reduce the thickness of the boundary lines by "peeling" their outer layers while preserving the distinctness of regions separated by them.

3 CALLING SEQUENCE

```
CALL PEELS (NTAPI, NTAPO, NREC, NEL, IT, MPASS, MDEV,
            NDEV, LX, LY, IBDY)
```

where

NTAPI, NTAPO are the logical unit numbers of the input and output sequential data sets;

NREC, NEL are the number of records and the number of pixels (bytes) per record in the input image;

IT is a threshold on density; if IT is positive (negative) all points with densities ≥ IT (≤ IT) will be regarded as boundary points;

MPASS is the maximum number of iterations permitted (see Section 9. Method);

MDEV, NDEV are logical unit numbers of two direct access scratch data sets defined as indicated in the listing of PEELS;

LX, LY, IBDY are scratch arrays with LX, LY dimensioned as indicated in the listing and IBDY dimensioned NEL.

4 INPUT-OUTPUT

4.1 Input

The input image should be on a sequential data set with unit number NTAPI and consist of NREC records and NEL bytes per record, each record corresponding to a line of the digitized image and each byte, to a pixel. All other inputs are as indicated in the calling sequence.

4.2 Output

The output of this program will be on unit NTAPO as a sequential data set with NREC records. The records will be in SLIC (scan line intersection code) format. That is, the first word of the I'th record indicates the number of words that follow and each subsequent word is a column coordinate of the intersection of the I'th scan line with the boundary image.

4.3 File Storage

This program requires two direct access scratch data sets to handle the intermediate iterations of the boundary data. The sizes of these data sets are indicated in the listings attached.

5 EXITS

No nonstandard exits.

6 USAGE

The program is in FORTRAN IV and implemented on the IBM 360 with the H compiler. The program is in the user's library as a load module.

7 EXTERNAL INTERFACES

This subroutine calls several subroutines and the linkage is shown in the following table.

8 PERFORMANCE SPECIFICATIONS

8.1 Storage

The subroutine PEELS is 1458 bytes long. However, including a driver (whose size depends largely on the dimensions of LX, LY, IBDY which are functions of NEL), the required subroutines and the buffers the program needs approximately 70K for handling NEL =2100.

8.2 Execution Time

The execution time is highly dependent on the size and complexity of the boundary image, the thickness of the boundary lines and the maximum number of passes (MPASS) requested. In the case of the Mobile Bay GTM (a 4000 x 2100 level II map with boundaries 3 and 4 pixels thick) the initial thresholding and reformatting took about 10 minutes and the subsequent iterations about 6 minutes each, with a final reformatting and copying step taking about 7 minutes. Thus, with MPASS=4, it takes about 40 minutes of CPU time to process the image.

Calling Program	Program Called
PEELS	PET SARN* VLTHR CMPRES DAWN* PEELER DARN EXPBDY
CMPRES	ISTORE+
PEELER	SVSCI PEELR1 PEELRO DAWN*
EXPBDY	ILOAD+
PEELR1	DARN BLSFTV BRSFTV•
PEELRO	IOR+ ICOMP1+ IAND+ BLSFTV
BLSFTV	ILOAD+ ISTORE+
BRSFTV	ILOAD+ ISTORE+

* Entry under DARN

\+ Logical function available in the user's library under a main member name LOGFUNC

• Entry under BLSFTV

8.3 Restrictions

None

9 METHOD

The program has three major steps:

(i) Thresholding, compressing and writing on a direct access unit.
(ii) Iterating to "peel" boundaries.
(iii) Changing to SLIC format and writing on output sequential data set.

9.1 Thresholding and Compressing

The routine SARN reads each record (of NEL bytes) of the input data set into the array LX. The routine VLTHR thresholds each of the NEL bytes in LX. A logical vector LY is defined as follows:

```
IF (IT.GE.0)LY(I) = LX(I).GE.IT
IT (IT.LT.0)LX(I) = LX(I).LE.IABS(IT)
```

for I = 1, NEL.

The routine CMPRES is then used to pack the information in LY into the first NEL bits of the array LX. The I'th bit of LX is "set" if and only if LY (I) is .TRUE..

The compressed boundary information is then written on the direct access unit MDEV using the routine DAWN.

9.2 Iterating to Peel

The main peeling routine is called PEELER. The input to this routine is from MDEV whenever IPASS, the iteration number, is odd and the output then will be written on NDEV. When IPASS is even, the input and output designations are interchanged. One call to PEELER removes one "layer" of the thick boundaries from top, left, bottom and right.

To decide whether a particular boundary point should be deleted (i.e. the bit corresponding to it changed to 0), we examine a 3x3 neighborhood centered around the point. Consider the array

```
a b c
d e f
g h i
```

where each letter represents a binary pixel. It is to be decided whether e, which is presently equal to 1 should be changed to 0. The conditions for a 'top peel' will be derived below and those for peeling from the other directions follow by symmetry.

First of all, e should be a top boundary point. That is, there should be no boundary point directly above e and there should be a boundary point below e. Therefore b=0 and h=1 are necessary conditions. Suppose $\bar{b}$ h=1. (Here, $\bar{b}$ denotes the complement of b). Then, we need only check whether e is a nonessential boundary point, that is, whether two 0's in the 3x3 array which are disconnected will stay disconnected where e is made 0. Connectivity, in this context, is defined as the existence of a path not including 1's and consisting only of horizontal and vertical segments.

Now, it is easy to see that e is essential if and only if $a\bar{d} = 1$ or $\bar{c}f = 1$. Therefore, the condition for a top peel is that

$$\bar{b}h\,(\bar{a}+d)\,(\bar{c}+f) = 1.$$

Equivalently, to perform a top peel we set

$$e = e\,(b+\bar{h}+a\bar{d}+c\bar{f}).$$

It is convenient to implement the above equation by employing bit manipulation routines operating on pairs of 32 bit words, thereby performing the top-peel operation in parallel on 32 pixels. This is done by using the "current" array in place of e, the "previous" array for b, the "next" array in place of h. Also, the previous, current, and next arrays are right (left) shifted by one bit and used for a, d and g (c, f and i) respectively in the peeling formulas.

The routine PEELER minimizes the movement of data in core by using circular buffers for storing the "previous, current and next" arrays. An array J dimensioned 3 is used to store the indices pointing to these arrays (J(1) ⟶ previous, J(2) ⟶ current, J(3) ⟶ next) and after finishing each record, only the array J is updated.

Also, top, left, bottom and right peels are performed one after the other by just one pass through the data (thus minimizing I/O) by storing the intermediate results in core and operating with a phase lag.

When the I'th record LX is read from the input data set (see PEELR1), BLSFTV and BRSFTV are used to generate arrays LXL and LXR with the bits in LX shifted by one bit to the left and right, respectively. Next, the (I-1)th record is peeled from the top. The top-peeled output of the (I-2)nd record is peeled from the left. The top-and left-peeled output of the (I-3)rd record is peeled from the bottom. The top-, left- and bottom- peeled output of the (I-4)th record is right-peeled and written on the output data set. Also, whenever any peeling is done other

than from the right the output is shifted to the left and right by one bit and the results are stored in the appropriate core locations pointed by J(3), K+ 1.

The routine PEELRO with the appropriate ISIDE will perform the peeling of one record. The above operations performed for I = 1, NREC+4 will complete one iteration of peeling, constituting one call to PEELER. The number, NP, of words of input that were changed is counted during each call to PEELER. If NP = 0 or the number of calls to PEELER has been MPASS, the iterations are stopped.

9.3 Converting to SLIC

Each record is read from the last scratch unit on which the output image was created. The routine EXPBDY is used to sense each bit in the record. The bit number of each 1-bit is stored in IBDY. The total number, N, of 1-bits followed by N words of the array IBDY are written on unit NTAPO.

10 COMMENTS

On large images this program takes a long time to execute. To avoid loss of data on long runs it is suggested that the direct access data sets be saved so that, with slight modifications, the routine PEELS can continue where the last run stopped due to insufficient CPU time.

11 LISTINGS

The listings of PEELS and most of the associated routines are attached at the end of this section. The routines not included are: PET, a routine used for printing time elapsed between sections of a program; SVSCI, a routine which sets all elements of an array to a given constant; DARN and the associated entry points for array read/write and the logical functions under member name LOGFUNC.

12 TESTS

The program was tested on a small portion of a boundary image, the image printed before and after peeling and was found to work satisfactorily.

```
ISN 0002          SUBROUTINE PEELS(NTI,NTO,NREC,NEL,IT,NPASS,MDEV,NDEV,LX,LY,IBDY)
ISN 0003          DIMENSION LX(1),LY(1),IBDY(1)
            C
            C     DIMENSION LX(36*((NEL-1)/32+1)),LY((NEL-1)/4+1)
            C     DEFINE FILE MDEV(NREC,(NEL-1)/32+1,U,IAV1)
            C     DEFINE FILE NDEV(NREC,(NEL-1)/32+1,U,IAV2)
            C
ISN 0004          N=(NEL-1)/32+1
ISN 0005          CALL PET(2)
ISN 0006          DO 10 I=1,NREC
ISN 0007          CALL SARN(NTI,LX,NEL)
ISN 0008          CALL VLTHR(LX,NEL,IT,LY)
ISN 0009          LX(N)=0
ISN 0010          CALL CMPRES(LY,NEL,LX)
ISN 0011    10    CALL DAWN(MDEV,I,LX,N*4)
ISN 0012          CALL PET(2)
ISN 0013          DO 20 IPASS=1,NPASS
ISN 0014          IF(MOD(IPASS,2).EQ.1)CALL PEELER(MDEV,NDEV,NREC,N,LX,LX(12*N+1),
                 LX(24*N+1),LY,NP)
ISN 0016          IF(MOD(IPASS,2).EQ.0)CALL PEELER(NDEV,MDEV,NREC,N,LX,LX(12*N+1),
                 LX(24*N+1),LY,NP)
ISN 0018          PRINT 100,IPASS,NP
ISN 0019          CALL PET(2)
ISN 0020          IF(NP.EQ.0)GO TO 30
ISN 0022    20    CONTINUE
ISN 0023          IPASS=NPASS
ISN 0024    30    JDEV=NDEV
ISN 0025          IF(MOD(IPASS,2).EQ.0)JDEV=MDEV
ISN 0027          DO 40 I=1,NREC
ISN 0028          CALL DARN(JDEV,I,LX,N*4)
ISN 0029          CALL EXPBDY(LX,N,NEL,IBDY,J)
ISN 0030    40    WRITE(NTO)J,(IBDY(L),L=1,J)
ISN 0031          CALL PET(2)
ISN 0032          RETURN
ISN 0033    100   FORMAT(5X'DURING PASS NUMBER'I3,' THROUGH PEELER'I6,' WORDS OF COM
                 -PRESSED BOUNDARY INFORMATION WERE CHANGED.')
ISN 0034          END
```

```
ISN 0002          SUBROUTINE VLTHR(LX,N,IT,LY)
ISN 0003          LOGICAL*1 LX(N),LY(N),F/.FALSE./,T/.TRUE./
            C
            C     THRESHOLD A VECTOR LX OF 8 BIT INTEGERS TO GET A T-F VECTOR.
            C     IF IT.GE.0, LX(I).GE.IT IMPLIES LY(I)=T.  IF IT<0 LX(I).LE.IABS(IT)
            C     IMPLIES LY(I)=T.
            C
ISN 0004          ITT=IABS(IT)
ISN 0005          IF(IT.LT.0)GO TO 10
ISN 0007          DO 20 I=1,N
ISN 0008          LY(I)=F
ISN 0009    20    IF(LX(I).GE.ITT)LY(I)=T
ISN 0011          RETURN
ISN 0012    10    DO 30 I=1,N
ISN 0013          LY(I)=F
ISN 0014    30    IF(LX(I).LE.ITT)LY(I)=T
ISN 0016          RETURN
ISN 0017          END
```

```
ISN 0002          SUBROUTINE CMPRES(LX,NEL,LY)
ISN 0003          LOGICAL*1 LX(NEL)
ISN 0004          DIMENSION LY(1)
ISN 0005          JWRD=1
ISN 0006          JBIT=33
ISN 0007          DO 10 I=1,NEL
ISN 0008          JBIT=JBIT-1
ISN 0009          IF(JBIT.NE.0)GO TO 20
ISN 0011          JBIT=32
ISN 0012          JWRD=JWRD+1
ISN 0013     20   IX=LX(I)
ISN 0014          LY(JWRD)=ISTORE(IX,LY(JWRD),JBIT,1)
ISN 0015     10   CONTINUE
ISN 0016          RETURN
ISN 0017          END
```

```
ISN 0002          SUBROUTINE PEELER(MDEV,NDEV,NREC,N,LX,LXR,LXL,LY,NP)
ISN 0003          DIMENSION LX(N,3,4),LXR(N,3,4),LXL(N,3,4),LY(N),J(3)
ISN 0004          NREC1=NREC+1
ISN 0005          NREC2=NREC+2
ISN 0006          NREC3=NREC+3
ISN 0007          NREC4=NREC+4
ISN 0008          J(1)=1
ISN 0009          J(2)=2
ISN 0010          J(3)=3
ISN 0011          CALL SVSCI(LX,N*12,0)
ISN 0012          CALL SVSCI(LXR,12*N,0)
ISN 0013          CALL SVSCI(LXL,12*N,0)
ISN 0014          NP=0
ISN 0015          DO 10 I=1,NREC4
ISN 0016          DO 20 K=1,4
ISN 0017          IF(I.LE.NREC+K)GO TO 20
ISN 0019          CALL SVSCI(LX(1,J(3),K),N,0)
ISN 0020          CALL SVSCI(LXR(1,J(3),K),N,0)
ISN 0021          CALL SVSCI(LXL(1,J(3),K),N,0)
ISN 0022     20   CONTINUE
ISN 0023          IF(I.LE.NREC)CALL PEELRI(MDEV,I,LY,J,N,LXR,LXL)
ISN 0025          IF(I.GT.1.AND.I.LE.NREC1)
                 .   CALL PEELRO(LX(1,1,1),LXR(1,1,1),LXL(1,1,1),J,N,1,
                 .     LX(1,J(3),2),LXR(1,J(3),2),LXL(1,J(3),2),NP)
ISN 0027          IF(I.GT.2.AND.I.LE.NREC2)
                 .   CALL PEELRO(LX(1,1,2),LXR(1,1,2),LXL(1,1,2),J,N,2,
                 .     LX(1,J(3),3),LXR(1,J(3),3),LXL(1,J(3),3),NP)
ISN 0029          IF(I.GT.3.AND.I.LE.NREC3)
                 .   CALL PEELRO(LX(1,1,3),LXR(1,1,3),LXL(1,1,3),J,N,3,
                 .     LX(1,J(3),4),LXR(1,J(3),4),LXL(1,J(3),4),NP)
ISN 0031          IF(I.GT.4)
                 .   CALL PEELRO(LX(1,1,4),LXR(1,1,4),LXL(1,1,4),J,N,4,
                 .     LY,0,0,NP)
ISN 0033          IF(I.GT.4)CALL DAWN(NDEV,I-4,LY,4*N)
ISN 0035          DO 30 K=1,3
ISN 0036     30   J(K)=MOD(J(K),3)+1
ISN 0037     10   CONTINUE
ISN 0038          RETURN
ISN 0039          END
```

```
ISN 0002          SUBROUTINE PEELRO(LX,LXR,LXL,J,N,ISIDE,LY,LYR,LYL,NP)
ISN 0003          DIMENSION LX(N,3),LXR(N,3),LXL(N,3),LY(N),J(3),IW(3),LYR(N),LYL(N)
ISN 0004          DO 60 I=1,N
ISN 0005          LY(I)=LX(I,J(2))
ISN 0006          IF(LY(I).EQ.0)GO TO 60
ISN 0008          GO TO (10,20,30,40),ISIDE
ISN 0009    10    IW(1)=IOR(LX(I,J(1)),ICOMP1(LX(I,J(3)),32,32))
ISN 0010          IW(2)=IAND(LXR(I,J(1)),ICOMP1(LXR(I,J(2)),32,32))
ISN 0011          IW(3)=IAND(LXL(I,J(1)),ICOMP1(LXL(I,J(2)),32,32))
ISN 0012          GO TO 50
ISN 0013    20    IW(1)=IOR(LXR(I,J(2)),ICOMP1(LXL(I,J(2)),32,32))
ISN 0014          IW(2)=IAND(LXR(I,J(1)),ICOMP1(LX(I,J(1)),32,32))
ISN 0015          IW(3)=IAND(LXR(I,J(3)),ICOMP1(LX(I,J(3)),32,32))
ISN 0016          GO TO 50
ISN 0017    30    IW(1)=IOR(LX(I,J(3)),ICOMP1(LX(I,J(1)),32,32))
ISN 0018          IW(2)=IAND(LXR(I,J(3)),ICOMP1(LXR(I,J(2)),32,32))
ISN 0019          IW(3)=IAND(LXL(I,J(3)),ICOMP1(LXL(I,J(2)),32,32))
ISN 0020          GO TO 50
ISN 0021    40    IW(1)=IOR(LXL(I,J(2)),ICOMP1(LXR(I,J(2)),32,32))
ISN 0022          IW(2)=IAND(LXL(I,J(1)),ICOMP1(LX(I,J(1)),32,32))
ISN 0023          IW(3)=IAND(LXL(I,J(3)),ICOMP1(LX(I,J(3)),32,32))
ISN 0024    50    IW(1)=IOR(IW(1),IW(2))
ISN 0025          IW(1)=IOR(IW(1),IW(3))
ISN 0026          LY(I)=IAND(LY(I),IW(1))
ISN 0027          IF(LX(I,J(2)).NE.LY(I))NP=NP+1
ISN 0029    60    CONTINUE
ISN 0030          IF(ISIDE.EQ.4)RETURN
ISN 0032          CALL BLSFTV(LY,N,LYL)
ISN 0033          CALL BRSFTV(LY,N,LYR)
ISN 0034          RETURN
ISN 0035          END
```

```
ISN 0002          SUBROUTINE BLSFTV(IX,N,IY)
ISN 0003          DIMENSION IX(N),IY(N)
ISN 0004          N1=N-1
ISN 0005          DO 10 I=1,N1
ISN 0006          IY(I)=ILOAD(IX(I+1),32,1)
ISN 0007    10    IY(I)=ISTORE(IX(I),IY(I),32,31)
ISN 0008          IY(N)=0
ISN 0009          IY(N)=ISTORE(IX(N),IY(N),32,31)
ISN 0010          RETURN
ISN 0011          ENTRY BRSFTV(IX,N,IY)
ISN 0012          IY(1)=ILOAD(IX(1),32,31)
ISN 0013          DO 20 I=2,N
ISN 0014          IY(I)=ILOAD(IX(I),32,31)
ISN 0015    20    IY(I)=ISTORE(IX(I-1),IY(I),32,1)
ISN 0016          RETURN
ISN 0017          END
```

```
ISN 0002          SUBROUTINE PEELRI(NDEV,I,LX,J,N,LXR,LXL)
ISN 0003          DIMENSION LX(N,3),LXR(N,3),LXL(N,3),J(3)
ISN 0004          CALL DARN(NDEV,I,LX(1,J(3)),N*4)
ISN 0005          CALL BLSFTV(LX(1,J(3)),N,LXL(1,J(3)))
ISN 0006          CALL BRSFTV(LX(1,J(3)),N,LXR(1,J(3)))
ISN 0007          RETURN
ISN 0008          END
```

```
ISN 0002          SUBROUTINE EXPBDY(LX,N,NEL,IBDY,J)
ISN 0003          DIMENSION LX(N),IBDY(1)
ISN 0004          LOGICAL ILOAD
ISN 0005          JWRD=1
ISN 0006          JBIT=33
ISN 0007          J=0
ISN 0008          DO 10 I=1,NEL
ISN 0009          JBIT=JBIT-1
ISN 0010          IF(JBIT.NE.0)GO TO 20
ISN 0012          JBIT=32
ISN 0013          JWRD=JWRD+1
ISN 0014      20  IF(.NOT.ILOAD(LX(JWRD),JBIT,1))GO TO 10
ISN 0016          J=J+1
ISN 0017          IBDY(J)=I
ISN 0018      10  CONTINUE
ISN 0019          RETURN
ISN 0020          END
```

5-5-2 FINDING DISCONTINUITIES IN BOUNDARY DATA

1 NAME

BOUDIM

2 PURPOSE

To find the discontinuities in digital curves stored in SLIC format.

3 CALLING SEQUENCE

CALL BOUDIM (IBDY, NTAPI, NREC, NEL, IRC, ND, NDIS)

where

IBDY is a scratch array to be dimensioned NEL*3 where NEL is the maximum number of boundary points in a given line;

NTAPI is the logical unit number on which the input boundary data are stored;

NREC is the number of lines (records) in the input data set;

IRC is the output array of coordinates of the discontinuities;

ND is the maximum number of discontinuities expected [IRC is dimensioned (ND, 2)];

NDIS is the output integer giving the actual number of discontinuities found.

NTAPI, NREC, NEL, ND are inputs to this routine IRC, NDIS are outputs.

4 INPUT-OUTPUT

4.1 Input

The input data should be on logical unit NTAPI as a sequential data set consisting of NREC records. Each record should consist of the coordinates of the intersection of the corresponding scan line with the boundary image written as

J, (X(I), I = 1, J)

where J is the number of such intersections and IX(I) are the coordinates.

4.2 Output

The output of this program is only through the calling sequence.

4.3 File Storage

None

5 EXITS

No nonstandard exits.

6 USAGE

This program is written in FORTRAN IV and is implemented on the IBM 360 with the H compiler. It is available on the users' library as a load module.

7 EXTERNAL INTERFACES

The linkage with other subroutines needed with this routine is indicated in the following table.

Calling Program	Program(s) Called
BOUDIM	BOUDIS
BOUDIS	JCOUNT

8 PERFORMANCE SPECIFICATIONS

8.1 Storage

This program is 834 bytes long. Including the external references listed above, the storage needed will be 2578 bytes (excluding the calling program which should provide storage for the arrays IRC and IBDY).

8.2 Execution Time

TBD

8.3 I/O Load

None

8.4 Restrictions

None

9 METHOD

The routine BOUDIM simply handles the I/O needed for finding the discontinuities. Connectivity, in this context, is defined in terms of the eight nearest neighbors of the point under consideration. Therefore, while examining the ith record of data, it is necessary to have the (i-1)st and (i+1)st records in core. The movement of data in core is avoided by using a circular buffer IBDY **dimensioned** (NEL, 3) and indexed by the pointer array IND dimensioned 3. Initially, IND is set to {1, 2, 3}. Always, IND(2) points to the current row. The numbers of boundary points in the three rows stored in core are in (NB(IND(J)), J = 1, 3). The routine BOUDIM starts by reading the first record into IBDY (*, 2). Then, for I = 1, NREC the (I+1)st record is read into IBDY (*, IND(3)). The (NREC+1)th record is undefined. Therefore, in that case, NB (IND(3)) is simply set to 0. The routine BOUDIS is called to determine the coordinates of the discontinuities on the I'th record. Then the pointer array IND is updated.

The functioning of BOUDIS is as follows. Each of the boundary points in the current record is treated as the point e in the following array.

a	b	c
d	e	f
g	h	i

The number of boundary points in this array excepting e is called the connectivity count of e. The connectivity count is calculated by examining the arrays IBDY (*, IND(2)), IBDY (*, IND(1)) and IBDY (*, IND(3)), stopping the calculations when the count equals 2. If the count is smaller than 2, then the point e is a discontinuity. The row and column coordinates of e and the continuity count are then stored in (IRC(NDIS,K), K = 1, 3).

10 COMMENTS

None

11 LISTINGS

The listings of this routine, with BOUDIS and JCOUNT are attached at the end.

12 TESTS

This program has been tested in conjunction with SMOB2, a smoothing routine documented in the next section.

```
ISN 0002              SUBROUTINE BOUDIN(IBDY,NTAPI,NREC,NEL,IRC,ND,NDIS)
                C
                C     FIND DISCONTINUITIES ON BOUNDARIES GIVEN THE INFO. ON NTAPI IN SLIC
                C     FORMAT.
                C
ISN 0003              DIMENSION IBDY(NEL,3),IND(3),NB(3)
ISN 0004              DIMENSION IRC(ND,2)
ISN 0005              DO 10 I=1,3
ISN 0006              IND(I)=I
ISN 0007        10    NB(I)=0
ISN 0008              READ(NTAPI)NB2,(IBDY(J,2),J=1,NB2)
ISN 0009              NB(2)=NB2
ISN 0010              NDIS=0
ISN 0011              DO 20 I=1,NREC
ISN 0012              IF(I.LT.NREC)READ(NTAPI)NB3,(IBDY(J,IND(3)),J=1,NB3)
ISN 0014              IF(I.EQ.NREC)NB3=0
ISN 0016              NB(IND(3))=NB3
ISN 0017              CALL BOUDIS(IBDY,IND,NB,NEL,I,NDIS,IRC,ND)
ISN 0018              DO 30 J=1,3
ISN 0019        30    IND(J)=MOD(IND(J),3)+1
ISN 0020        20    CONTINUE
ISN 0021              RETURN
ISN 0022              END




ISN 0002              SUBROUTINE BOUDIS(IBDY,IND,NB,NEL,IR,NDIS,IRC,ND)
ISN 0003              DIMENSION IBDY(NEL,?),IND(3),NB(3)
ISN 0004              DIMENSION IRC(ND,3)
                C
                C     IBDY(J,IND(I)),J=1,NB(IND(I)) ARE THE BOUNDARY COORDINATES IN THE
                C     PREVIOUS, CURRENT AND NEXT LINES FOR I=1,2,3 RESPECTIVELY.
                C     FIND THE DISCONTINUITIES AT THE CURRENT LINE.  A DISCONTINUITY
                C     IS DEFINED AS A BOUNDARY POINT NOT CONNECTED TO AT LEAST TWO OTHER
                C     BOUNDARY POINTS.
                C     IT IS ASSUMED THAT THE BOUNDARY POINTS IN EACH ROW ARE IN ASCEND-
                C     ING ORDER.
                C
ISN 0005              NB1=NB(IND(1))
ISN 0006              NB2=NB(IND(2))
ISN 0007              NB3=NB(INC(3))
ISN 0008              IF(NB2.EQ.0)RETURN
ISN 0010              DO 10 J=1,NB2
ISN 0011              ICOUNT=0
ISN 0012              IF(J.GT.1.AND.IBDY(J,IND(2))-IBDY(J-1,IND(2)).EQ.1)ICOUNT=
                     .      ICOUNT+1
ISN 0014              IF(J.LT.NB2.AND.IBDY(J+1,IND(2))-IBDY(J,IND(2)).EQ.1)ICOUNT=
                     .      ICOUNT+1
ISN 0016              IF(ICOUNT.GE.2)GO TO 10
ISN 0018              IF(NB1.NE.0)ICOUNT=
                     .  ICOUNT+JCOUNT(IBDY,(IND(2)-1)*NEL+J,(IND(1)-1)*NEL+1,(IND(1)-1)
                     .      *NEL+NB1)
ISN 0020              IF(ICOUNT.GE.2)GO TO 10
ISN 0022              IF(NB3.NE.0)ICOUNT=ICOUNT+
                     .  JCOUNT(IBDY,(IND(2)-1)*NEL+J,(IND(3)-1)*NEL+1,(IND(3)-1)*NEL+NB3
                     .      )
ISN 0024              IF(ICOUNT.GE.2)GO TO 10
ISN 0026              NDIS=NDIS+1
ISN 0027              WRITE(6,100)NDIS,IR,IBDY(J,IND(2)),ICOUNT
ISN 0028              IRC(NDIS,1)=IR
ISN 0029              IRC(NDIS,2)=IBDY(J,IND(2))
ISN 0030              IRC(NDIS,3)=ICOUNT
ISN 0031        10    CONTINUE
ISN 0032              RETURN
ISN 0033       100    FORMAT(' DISCONTINUITY NO.'I5,' AT ('I4,','I4,'), ICOUNT='I2,'.')
ISN 0034              END
```

```
ISN 0002          FUNCTION JCOUNT(IX,J,J1,J2)                                00000000
ISN 0003          DIMENSION IX(1)                                            00000010
            C     JCOUNT= NO. OF VALUES OF JJ SUCH THAT J1.LE.JJ.LE.J2       00000030
            C     AND IABS(IX(J)-IX(JJ)).LE.1                                00000040
ISN 0004          JCOUNT=0                                                   00000050
ISN 0005          IF(J1.GT.J2)RETURN                                         00000060
ISN 0007          K=0                                                        00000070
ISN 0008          DO 10 JJ=J1,J2                                             00000080
ISN 0009          IF(IX(JJ)-IX(J).LT.-1)GO TO 10                             00000090
ISN 0011          IF(IX(JJ)-IX(J).GT.1)GO TO 20                              00000100
ISN 0013          K=K+1                                                      00000110
ISN 0014    10    CONTINUE                                                   00000120
ISN 0015    20    JCOUNT=K                                                   00000130
ISN 0016          RETURN                                                     00000140
ISN 0017          END                                                        00000150
```

5-5-3 SMOOTHING BOUNDARY DATA

1 NAME

SMOB2

2 PURPOSE

To patch discontinuities in a digital curve.

3 CALLING SEQUENCE

CALL SMOB2(IRC, MDIS, IDIS, NDIS, NDEV, IBDY, IW1, IW2, NREC, K)

where

IRC is an input array dimensioned (MDIS, 3) with IRC(I, 1), IRC(I, 2)) giving the row and column coordinates of the I'th discontinuity and IRC(I, 3) giving its connectivity count for I = 1 through NDIS;

IDIS is the discontinuity number at which the patching should be started (only the discontinuities corresponding to I = IDIS through NDIS will be patched);

NDEV is the logical unit number of a direct access device on which the input boundary data set is located; the output after smoothing is written back on NDEV.

IBDY, IW1, IW2 are work arrays to be dimensioned as indicated in the listing attached;

NREC is the number of records in the boundary image;

K is maximum coordinate difference over which the nearest boundary points are checked for patching a discontinuity. (See 9, Method).

All parameters except the work arrays are inputs.

4 INPUT-OUTPUT

4.1 Input

The input data should be on the direct access unit NDEV, consisting of NREC records, the I'th record readable by

READ(NDEV'I)N, (IBDY(J, 1)), J = 1, N).

4.2 Output

The output data will be on NDEV in the same format as the input.

4.3 File Storage

None.

5 EXITS

No nonstandard exits.

6 USAGE

The program is in FORTRAN IV and is presently implemented on IBM 360 using the H compiler. It is available on the user's library in the form of a load module.

7 EXTERNAL REFERENCES

The linkage is indicated in the following table:

Calling Program	Programs Called
SMOB2	PATCH3
PATCH3	SVSCI PATCH1 SORT ELIRPT
PATCH1	CONTEL PATCH4 PATCH2 PRTVEC
SORT	MVMRMR
ELIRPT	VMOV

8 PERFORMANCE SPECIFICATIONS

8.1 Storage

The size of SMOB2 is 1068 bytes. Including a main program to supply the arrays required to handle a maximum of 2100 boundary points per record with K = 20 and the buffers, this program needs approximately 114K bytes for execution.

8.2 Execution Time

Highly dependent on the image size, complexity and the number of discontinuities to be patched. In the case of the Mobile Bay, Alabama level II GTM which had 4000 records with 728 discontinuities of which about 530 required patches to be generated, the execution time on IBM 360/65 was about 9 minutes. Since there is a considerable amount of I/O involved on the direct access unit NDEV, a significant improvement in execution time can be achieved by using the array read/write routines DARN and DAWN wherever implied DO loops have been used in the subroutine PATCH3.

8.3 I/O Load

None

8.4 Restrictions

None

9 METHOD

The routine SMOB2 simply consists of a DO loop which calls PATCH3 to generate the patch points needed for the L'th discontinuity and prints the details of the patches produced, with L ranging from IDIS through NDIS.

Consider the routine PATCH3. Suppose (I, J) is the address of the discontinuity at which a patch is to be produced. Then, the records I-K through I+K (bounded, of course, by 1 and NREC) are read from NDEV. While each record is read one row of a 2K+1 by 2K+1 binary matrix IW1 is defined. The elements of the row are initially set to 0 and whenever the (J-K+L)'th column in the present row of the input image has a boundary point, the (L+1)st element is set to 1.

After defining IW1, the routine PATCH1 is used to check the array IW1, eliminate the 1's contiguous with the (K+1, K+1)th element, find the nearest 1 among the remaining and join it to that element by a straight line and store the row and column coordinates of the points so produced in an array IW2. Further, if the contiguity count of the point of interest is 0, then the 1's contiguous with

the point joined to the point of interest are also eliminated and a straight line patch is produced to the nearest remaining 1.

The addresses in IW2 are then merged with the data on the input direct access data set by reading the corresponding records of input, sorting the column coordinates in each record using SORT, eliminating repetitions of column coordinates using ELIRPT and writing back on NDEV.

10 COMMENTS

The routine SMOB2 can be used in conjunction with BOUDIM or independently. If used independently, the coordinates of discontinuities may be supplied by reading a sequential data set produced by a separate run of BOUDIM. If the program terminates due to lack of time, the execution can be continued by a subsequent run with an updated value of IDIS provided the output data set on NDEV is kept.

11 LISTINGS

Listings of SMOB2 and the important routines called by it are shown at the end of this section.

12 TESTS

This routine has been tested by using the coordinates of the discontinuities produced by BOUDIM on the Mobile Bay GTM. The first 40 discontinuities were examined in detail by printing the arrays IW1. The performance of the routine was found to be satisfactory.

```
ISN 0002              SUBROUTINE SMOO2(IRC,MDIS,IDIS,NDIS,NDEV,IBDY,IW1,IW2,NREC,K)
ISN 0003              COMMON/PTCHAD/I1,J1,I2,J2,NP
ISN 0004              DIMENSION IRC(MDIS,2),  IBDY(1)  ,IW1(1),IW2(1)
            C         D'N IBDY(MAX. EXPECTED NO. OF BOUNDARY POINTS IN A LINE AFTER SMOOTHING)
            C
            C         D'N IW1(K21**2),IW2(K21**2) WHERE K21=2*K+1.
            C
ISN 0005              DO 20 I=IDIS,NDIS
ISN 0006              CALL PATCH3(IBDY,IRC(I,1),IRC(I,2),IRC(I,3),K,IW1,IW2,NREC,NDEV)
ISN 0007              IF(I2.NE.0)PRINT 100,I,IRC(I,1),IRC(I,2),I1,J1,I2,J2
ISN 0009              IF(I1.NE.0.AND.I2.EQ.0)PRINT 101,I,IRC(I,1),IRC(I,2),I1,J1
ISN 0011              IF(I1.EQ.0)PRINT 102,I,IRC(I,1),IRC(I,2)
ISN 0013       20     CONTINUE
ISN 0014              RETURN
ISN 0015      100     FORMAT(2XI5,': (',I5,',',I5,') JOINED TO (',I5,',',I5,') AND (',I5,',',
                    .     I5,').')
ISN 0016      101     FORMAT(2XI5,': (',I5,',',I5,') JOINED TO (',I5,',',I5,').')
ISN 0017      102     FORMAT(2XI5,': NO PATCH POINTS PRODUCED AT (',I5,',',I5,').')
ISN 0018              END
```

```
ISN 0002              SUBROUTINE PATCH1(IW1,IW2,M,N,I,J,ICOUNT)
ISN 0003              COMMON/PTCHAD/I1,J1,I2,J2,NP
ISN 0004              DATA IPASS/0/
ISN 0005              IPASS=IPASS+1
ISN 0006              IF(IPASS.LE.40)CALL PRTVEC(IW1,M*N,1)
            C
            C         GENERATE PATCH POINTS IN IW1 STARTING FROM (I,J) TO THE NEAREST
            C         (2-ICOUNT) NONCONTIGUOUS NEIGHBORS.
            C
            C         SEE CONTEL FOR DIMENSIONING INFO FOR IW2.
            C
ISN 0008              DIMENSION IW1(M,N),IW2(2,1)
ISN 0009              IW2(1,1)=I
ISN 0010              IW2(2,1)=J
            C
            C         ELIMINATE POINTS CONTIGUOUS WITH (I,J).
            C
ISN 0011              CALL CONTEL(IW1,IW2,M,N)
            C
            C         FIND NEAREST NEIGHBOR OF (I,J) WHICH IS SET.
            C
ISN 0012              I2=0
ISN 0013              CALL PATCH6(IW1,M,N,I,J,I1,J1)
            C         NOW (I1,J1) IS THE NEAREST NEIGHBOR.
            C
ISN 0014              IF(ICOUNT.NE.0.OR.I1.EQ.0)GO TO 10
            C
            C         ELIMINATE POINTS CONTIGUOUS WITH (I1,J1)
            C
ISN 0016              IW2(1,1)=I1
ISN 0017              IW2(2,1)=J1
ISN 0018              CALL CONTEL(IW1,IW2,M,N)
ISN 0019              CALL PATCH6(IW1,M,N,I ,J ,I2,J2)
            C         NOW (I2,J2) IS THE NEXT NEAREST NEIGHBOR.
ISN 0020       10     CONTINUE
ISN 0021              MP=0
ISN 0022              NP=0
ISN 0023              IF(I1.EQ.0)RETURN
            C
            C         PRODUCE PATCH ADDRESSES IN IW2.
            C
ISN 0025              CALL PATCH2(IW2,NP,I1,J1,I,J)
ISN 0026              IF(I2.NE.0)CALL PATCH2(IW2(1,NP+1),MP,I2,J2,I,J)
ISN 0028              NP=NP+MP
ISN 0029              IF(IPASS.LE.50)CALL PRTVEC(IW2,2*NP,2)
ISN 0031              RETURN
ISN 0032              END
```

```
ISN 0002          SUBROUTINE PATCH2(IW,NP,I1,J1,I2,J2)
ISN 0003          DIMENSION IW(2,1)
            C
            C     TO GENERATE COORDINATES OF LINE JOINING (I1,J1) AND (I2,J2).
            C     DIMENSION IW(2,NP) WHERE NP= MAX. NO. OF POINTS EXPECTED TO BE
            C     PRODUCED.  NP= NO. OF POINTS ACTUALLY PRODUCED BY THIS ROUTINE.
            C
ISN 0004          IMN=MINO(I1,I2)+1
ISN 0005          IMX=MAXO(I1,I2)-1
ISN 0006          JMN=MINO(J1,J2)+1
ISN 0007          JMX=MAXO(J1,J2)-1
ISN 0008          I12=I1-I2
ISN 0009          J12=J1-J2
ISN 0010          RI12=I12
ISN 0011          RJ12=J12
ISN 0012          NP=0
ISN 0013          IF(IMX-IMN.GT.JMX-JMN)GO TO 10
ISN 0015          IF(JMN.GT.JMX)RETURN
ISN 0017          DO 20 J=JMN,JMX
ISN 0018          I=I1+(J-J1)*I12/RJ12+.5
ISN 0019          NP=NP+1
ISN 0020          IW(1,NP)=I
ISN 0021    20    IW(2,NP)=J
ISN 0022          RETURN
            C
ISN 0023    10    IF(IMN.GT.IMX)RETURN
ISN 0025          DO 30 I=IMN,IMX
ISN 0026          NP=NP+1
ISN 0027          J=J1+(I-I1)*J12/RI12+.5
ISN 0028          IW(1,NP)=I
ISN 0029    30    IW(2,NP)=J
ISN 0030          RETURN
ISN 0031          END
```

```
ISN 0002          SUBROUTINE CONTEL(IW1,IW2,M,N)
ISN 0003          DIMENSION IW1(M,N),IW2(2,1)
            C
            C     THIS PROGRAM ELIMINATES ALL 1'S IN IW1 CONNECTED TO THE 1 AT
            C     (IW1(1,1),IW2(2,1)).  IW2 SHD BE DIMENSIONED (2,K) WHERE K IS TWICE
            C     THE NUMBER OF NODES IN THE PIECE OF DIGITAL CURVE CONNECTED TO THE
            C     POINT OF INTEREST.
            C
ISN 0004          K=1
ISN 0005          L=1
ISN 0006    40    DO 20 KK=1,K
ISN 0007          I=IW2(1,KK)
ISN 0008          J=IW2(2,KK)
ISN 0009          IW1(I,J)=0
ISN 0010          I1=MAXO(I-1,1)
ISN 0011          I2=MINO(I+1,M)
ISN 0012          J1=MAXO(J-1,1)
ISN 0013          J2=MINO(J+1,N)
ISN 0014          DO 10 II=I1,I2
ISN 0015          DO 10 JJ=J1,J2
ISN 0016          IF(IW1(II,JJ).EQ.0)GO TO 10
ISN 0018          L=L+1
ISN 0019          IW2(1,L)=II
ISN 0020          IW2(2,L)=JJ
ISN 0021          IW1(II,JJ)=0
ISN 0022    10    CONTINUE
ISN 0023    20    CONTINUE
ISN 0024          IF(L.EQ.K)RETURN
ISN 0026          K1=K+1
ISN 0027          LL=0
ISN 0028          DO 30 KK=K1,L
ISN 0029          LL=LL+1
ISN 0030          IW2(1,LL)=IW2(1,KK)
ISN 0031    30    IW2(2,LL)=IW2(2,KK)
ISN 0032          K=LL
ISN 0033          L=LL
ISN 0034          GO TO 40
ISN 0035          END
```

```
ISN 0002          SUBROUTINE PATCH3(IBDY,I,J,ICOUNT,K,IW1,IW2,NREC,NDEX)
ISN 0003          COMMON/PTCHAD/I1,J1,I2,J2,NP
            C     DIMENSION IW1(K21**2),IW2(2,NP), WHERE NP=MAX(NO. OF PATCH POINTS
            C     EXPECTED TO BE GENERATED BY PATCH2, DIMENSION REQUIRED BY CONTEL)
            C     PATCH DISCONTINUITY AT I,J.
ISN 0004          DIMENSION IW1(1),IW2(2,1),IBDY(1)
ISN 0005          K21=K*2+1
ISN 0006          K1=MAX0(I-K,1)
ISN 0007          K2=MIN0(I+K,NREC)
ISN 0008          KK21=K2-K1+1
ISN 0009          CALL SYSC1(IW1,KK21*K21,0)
ISN 0010          ICLMMK=J-K
ISN 0011          ICLMPK=J+K
ISN 0012          DO 10 KK=K1,K2
ISN 0013          READ(NDEX'KK)N,(IBDY(L),L=1,N)
ISN 0014          IF(N.EQ.0)GO TO 10
ISN 0016          DO 20 L=1,N
ISN 0017          IF(IBDY(L).LT.ICLMMK)GO TO 20
ISN 0019          IF(IBDY(L).GT.ICLMPK)GO TO 10
            C
            C     TREAT IW1 AS A 2-D ARRAY DIMENSIONED KK21*K21 WITH THE GIVEN POINT
            C     AT (I-K1+1,K+1)'TH LOCATION.
            C
ISN 0021          IW1((IBDY(L)-ICLMMK)*KK21+KK-K1+1)=1
ISN 0022    20    CONTINUE
ISN 0023    10    CONTINUE
            C
            C     GENERATE PATCH ADDRESSES IN IW2.
            C
ISN 0024          CALL PATCH1(IW1,IW2,KK21,K21,I-K1+1,K+1,ICOUNT)
            C
            C     MERGE ADDRESSES FOUND IN IW2 WITH THE BOUNDARY ADDRESSES ON DISC.
            C
ISN 0025          IF(NP.EQ.0)RETURN
ISN 0027          IP=1
ISN 0028          IP1=1
ISN 0029    30    IP=IP+1
            C
            C     FIND NEXT CHANGE IN IW2(1,*)
            C
ISN 0030          IF(IP.GT.NP)GO TO 40
ISN 0032          IF(IW2(1,IP).EQ.IW2(1,IP-1))GO TO 30
ISN 0034    40    IP2=IP-1
ISN 0035          KK=IW2(1,IP2)+K1-1
ISN 0036          READ(NDEX'KK)N,(IBDY(L),L=1,N)
ISN 0037          DO 50 JP=IP1,IP2
ISN 0038          N=N+1
ISN 0039    50    IBDY(N)=IW2(2,JP)+ICLMMK-1
ISN 0040          CALL SORT(IBDY,1,N,N,1,T,TT)
ISN 0041          CALL ELIRPT(N,IBDY)
ISN 0042          WRITE(NDEX'KK)N,(IBDY(L),L=1,N)
ISN 0043          IP1=IP
ISN 0044          IF(IP.LE.NP)GO TO 30
ISN 0046          RETURN
ISN 0047          END
```

```
ISN 0002           SUBROUTINE PATCH6(IN1,M,N,I,J,ID,JD)
ISN 0003           DIMENSION IN1(M,N)
ISN 0004           ID=0
ISN 0005           JD=0
ISN 0006           IF(I.EQ.0)RETURN
ISN 0008           IDMIN=M**2+N**2+1
ISN 0009           DO 10 II=1,M
ISN 0010           DO 10 JJ=1,N
ISN 0011           IF(IN1(II,JJ).EQ.0)GO TO 10
ISN 0013           ID=(II-I)**2+(JJ-J)**2
ISN 0014           IF(ID.GE.IDMIN)GO TO 10
ISN 0016           ID=II
ISN 0017           JD=JJ
ISN 0018           IDMIN=ID
ISN 0019      10   CONTINUE
ISN 0020           RETURN
ISN 0021           END
```

```
ISN 0002           SUBROUTINE PRTVEC(IX,N,IFMT)
ISN 0003           DIMENSION IX(N)
ISN 0004           IF(IFMT.EQ.1)PRINT 100,IX
ISN 0006           IF(IFMT.EQ.2)PRINT 200,IX
ISN 0008           RETURN
ISN 0009     100   FORMAT(10X41I1)
ISN 0010     200   FORMAT(1X40I3)
ISN 0011           END
```

5-5-4 IDENTIFICATION OF CONNECTED REGIONS

1 NAME

REGIONS

2 PURPOSE

To identify all distinct connected regions in an image given the boundary data in SLIC format and produce a map with a number at each point showing the region to which it belongs. The region numbers will be in descending order of area.

3 CALLING SEQUENCE

This is a main program. In its present version the image size is supplied through DATA statements.

4 INPUT-OUTPUT

4.1 Input

The input to this program is a sequential data set on logical unit 8, having NREC records stored as N, (IX(J), J = 1, N) in unformatted FORTRAN mode.

4.2 Output

The output of this program will be a sequential data set on logical unit 12, having NREC records with NEL pixels each, with one half-word (2 bytes) per pixel.

4.3 File Storage

This program requires a direct access data set with NREC records and NEL half-words per record.

5 EXITS

Not applicable

6 USAGE

This program is in FORTRAN IV and is implemented on IBM 360 with the H compiler. The associated subroutines are available as load modules on the user's library. The deck for the main program is available with the authors and needs only slight modifications in the DEFINE FILE and DATA statements for use on any data set.

7 EXTERNAL INTERFACES

This program uses several subroutines as indicated by the linkage table below:

Calling Program	Programs Called
REGIONS	PET VMAXI4 VMINI4 RIDER SVSCI DARN SEQLS SAWN
RIDER	SVSCI2 SVSCL1 SORT RIDER1 RIDER4 DAWN VMOV2 RIDER2 PRTVE2 DARN VMAXI2
SEQLS	SORT FLIPV
SORT	MVMRMR
RIDER1	SVSCI2
RIDER4	RIDER5 SVSCI2 PRTVE2 RIDER7 RIDER6 SVSCL1

Calling Program	Programs Called
RIDER2	PRTVE2
RIDER7	VMAXI2 VMINI2

8 PERFORMANCE SPECIFICATIONS

8.1 Storage

The present version of the main program is 134,436 bytes long. The external references required and the buffers increase this to 192K bytes. However, the size is dependent on the data set to be handled and the dimension statements should be changed to satisfy specific requirements.

```
DIMENSION   IX(2NR+2,N), IRES(MSEG+1)
INTEGER*2   IW1(NEL), IW2(NEL), ITABL(MR*MSEG), IS(MR)
INTEGER*2   LW(MR)
LOGICAL*1   IDENT(MR,MR)
```

where

NR = Maximum number of regions expected;
N = Maximum number of boundary points expected in a record;
MSEG = Maximum number of "segments" required to handle the image (see 9, Method);
MR = Maximum number of region identifiers permissible in a segment;
NEL = Number of pixels per line in the output map.

8.2 Execution Time

The time is highly dependent on the size and complexity of the image. The Mobile Bay GTM (level II) resulting in a region identification map with 400x2100 pixels and consisting of 1742 regions had to be handled in 15 sections and took 19.5 minutes of CPU time on IBM 360/65.

8.3 Restrictions

None

9 METHOD

This program has five major sections.

(i) Determination of the bounds on the column coordinates of boundaries on the input data set;

(ii) Finding a preliminary set of region identifiers;

(iii) Finding the areas of each of the regions;

(iv) Generating a mapping such that the region numbers are used in the order of decreasing areas;

(v) Modifying the region numbers by table look-up.

9.1 Determination of Bounds

The maximum and minimum values of the column coordinates of the boundary points are determined. If the minimum is greater than 1, it is set to 1. If the maximum is less than the value of NEL supplied, it is set to NEL. The value of NEL is then changed to Max-Min+1. The output image size will then be NREC by NEL.

9.2 Finding Preliminary Region Identifiers

This is the most important step in the program. The subroutine RIDER is used for this purpose. Its function is similar to the routine with the same name described in [23]. The routine in [23] was designed to print an error message and return with NR=0 when the number of distinct regions exceeded MR. But the present version can handle up to MR*MSEG distinct regions while still using a "region identity matrix" of size MR by MR (rather than MR*MSEG by MR*MSEG).

This routine uses the arrays IW1 and IW2 as the previous and current records of region identifiers. By convention, region numbers 1 and 0 indicate the "exterior" of the image and boundary points. The MR by MR array IDENT is used to store information about identity of regions, IDENT(I, J) = .TRUE. meaning that region numbers I and J refer to the same connected region.

Initially, the array IW1 is set to all 1's and IDENT is set to all .FALSE.. Each of the input records is read and the following operations are performed.

The boundary coordinates in the input record are arranged in ascending order. The routine RIDER1 is used to generate, in IW2, the region identification numbers corresponding to the present row. First, all the elements of IW2 corresponding to the boundary coordinates are set to zero. Each interval between

the zeros is compared with the corresponding segment of IW1. If there is no nonzero element in that segment of IW1, a new region number is started and assigned to the interval in IW2. If there is a nonzero element, that number is filled into all elements in the interval. Finally, IDENT(IW1(I), IW2(I)) is set to .TRUE. for I=1, NEL wherever IW1(I) ≠ 0 and IW2(I) ≠ 0, indicating that IW1(I) and IW2(I) refer to the same region. Also, when new region identifiers are to be used, the routine RIDER1 verifies whether the number of identifiers exceeds MR. If so, the value of NR, the total number identifiers, is set to -NRP, the total number up to the previous record and the control goes back to the routine RIDER.

Now, if RIDER1 returns a positive NR, the array IW2 is written as the I'th record on the direct access data set (unit number IDEVO in RIDER, same as 90 in the main program) and IW2 is moved into IW1 (so that it becomes the "previous" record while handling the next record).

If RIDER1 returns a negative NR, then NR is changed to -NR and the routine RIDER4 is called. The set of records handled between any two calls of RIDER4 will be referred to as a segment. Associated with each segment, a table is defined which gives a mapping from the set of region identifiers obtained in that segment to a new set reflecting the connectivities discovered up to the most recent segment handled. Also, the initial record number for each of the segments is stored in an array. The functions of the routine RIDER4 are to:

(i) Reduce the matrix IDENT (using RIDER5) examining all of the available connectivity information in it and obtain a look-up table for the current segment;

(ii) Modify the tables for the previous segments to reflect the newly found connectivities, if any;

(iii) Find all the distinct region numbers occuring in the last record IW1 of the current segment and change the numbers there which are greater than 1 to consecutive numbers starting with 2; Let NR be the largest number in IW1;

(iv) Set up an array IS consisting of the distinct region numbers in IW1 and then change IS(I) to ITABL(IS(I), ISEG) where ITABL is the look-up table for the current segment;

(v) Set all elements of IDENT TO .FALSE. except wher IS(I) = IS(J) for I, J in the range 1 through NR.

After each call to RIDER4, the segment count ISEG is incremented and the initial record number for the next segment (which is really the record number at which RIDER4 had to be called) is stored in IRES(ISEG). If MSEG is exceeded by ISEG or if NR > MR (which means there are more than MR distinct regions in the last record) the routine RIDER prints an error message, sets NR = 0 and exits.

Otherwise, RIDER1 is called again, IW2 is found and written on IDEVO and the program proceeds normally to the next input record.

After the NREC input records have been processed the routine RIDER4 is called to get the look-up table for the final segment. A call to RIDER2 changes the look-up tables for all the segments such that consecutive region numbers are used.

Finally, each record from IDEVO is read, the appropriate look-up table is used to modify it and the record is written back on IDEVO. Also, NR is set to the maximum region number used after table look-up.

9.3 Finding Areas

A histogram of the region identification maps is found, giving the total number of occurrences of each of the region identifiers 0 through NR. These numbers indicate the areas of the regions.

9.4 Finding the Final Look-up Table

A sequence of natural numbers is used as a secondary array with the histogram as the primary array in a descending sort operation (routine SEQLS). The resulting secondary array then gives the sequence of original region identifiers corresponding to decreasing areas. An inverse mapping [inverse mapping of {IX(J) J = 1, N} is defined as {IY(J) J = 1, N} if IY (IX(J)) = J.] of this sequence gives the final look-up table. The actual coding follows these principles but is slightly different in detail to preserve the identities of regions 0 and 1 which have special significance.

9.5 Deriving the Final Region Identification Map

The look-up table generated above is used to modify the region identifiers on IDEVO, record by record, and write out the final sequential data set on unit 12.

10 COMMENTS

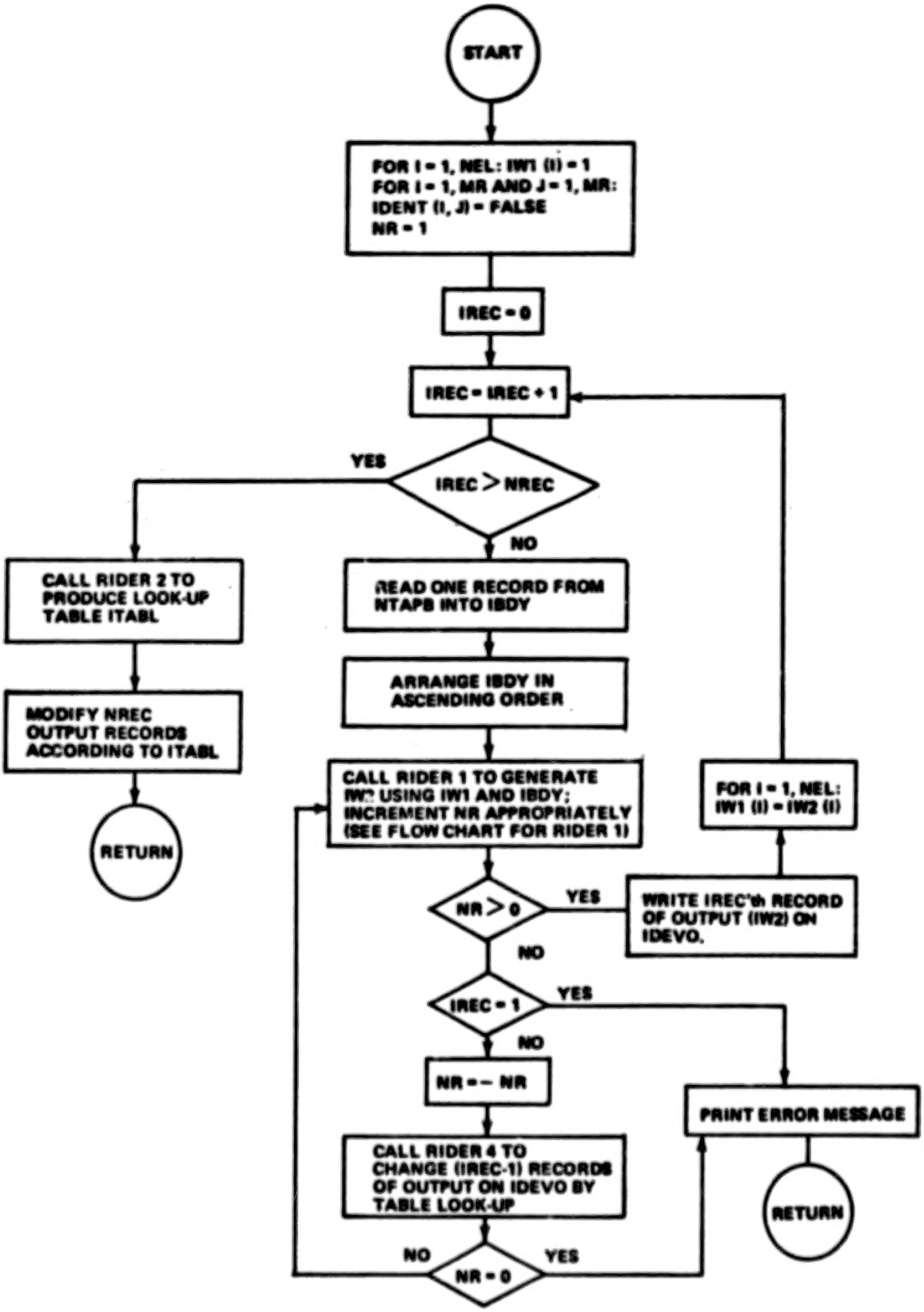
An approach suggested in [24] can be used instead of the one described above. With that method, the processing would be identical, except that the matrix IDENT is not defined. Instead, a table is updated every time a new connectivity is discovered. While this saves storage, it appears to take more execution time than the present method.

11 LISTINGS

The listings of the main program and the associated routines are attached at the end of this section.

12 TESTS

This program has been tested on the Mobile Bay GTM both before and after smoothing and found to work satisfactorily. Also, the results have been found to be identical (on a smaller data set) with those obtained by the earlier version of this program.



FLOW CHART FOR RIDER

```
ISN 0002          DIMENSION IX(4000),IRES(21)
ISN 0003          INTEGER*2 IW1(2100),IW2(2100),ITABL(8000),IS(400)
ISN 0004          INTEGER*2 LW(400)
ISN 0005          LOGICAL*1 IDENT(300,300)
ISN 0006          DEFINE FILE 90(4000,4200,L,IAV)
            C
            C     D'N IX(MAX(2NR+2,N) WHERE NR=MAX. NO. OF REGIONS EXPECTED AND
            C         N= MAX NO. OF BOUNDARY POINTS EXPECTED IN ANY RECORD)
            C
ISN 0007          DATA NREC,MR,MSEG/4000,300,20/
ISN 0008          DATA NEL/2100/
ISN 0009          MAXX=-1000000
ISN 0010          MINX= 1000000
ISN 0011          CALL STRTMR
ISN 0012          CALL PET(0)
ISN 0013          DO 10 I=1,NREC
ISN 0014          READ(8)N,(IX(J),J=1,N)
ISN 0015          IF(N.EQ.0)GO TO 10
ISN 0017          CALL VMAXI4(IX,N,MAXX)
ISN 0018          CALL VMINI4(IX,N,MINX)
ISN 0019     10   CONTINUE
ISN 0020          REWIND 8
            C     IDENTIFY CONNECTED REGIONS.
ISN 0021          PRINT 600,MINX,MAXX,NEL
ISN 0022          MINX=MINO(MINX,1)
ISN 0023          PRINT 600,MINX,MAXX,NEL
ISN 0024          MAXX=MAXO(MAXX,NEL)
ISN 0025          PRINT 600,MINX,MAXX,NEL
ISN 0026          NEL=MAXX-MINX+1
ISN 0027          PRINT 600,MINX,MAXX,NEL
ISN 0028    600   FORMAT(' MINX,MAXX,NEL='3I8)
ISN 0029          PRINT 100,NREC,NEL
ISN 0030    100   FORMAT(//' IMAGE SIZE=('I5,',',I5,')')
ISN 0031          NDUM=1
ISN 0032          PRINT 1000,NDUM
ISN 0033   1000   FORMAT(' NDUM='I5)
ISN 0034          CALL PET(2)
ISN 0035          CALL RIDER(8,NREC,NEL,90,MINX,IX,IW1,IW2,ITABL,IDENT,MR ,NR,LW,
                 *   MSEG,IRES,IS)
ISN 0036          CALL PET(2)
            C
            C     FIND AND PRINT HISTOGRAM OF REGION IDENTIFICATION MAP.
            C
ISN 0037          PRINT 200
ISN 0038    200   FORMAT(//10X'REGION NO.'10X'NO. OF PIXELS')
ISN 0039          CALL SVSCI(IX,NR+1,0)
ISN 0040          DO 30 I=1,NREC
ISN 0041          CALL DARN(90,I,IW1,NEL*2)
ISN 0042          DO 30 IEL=1,NEL
ISN 0043          J=IW1(IEL)+1
ISN 0044     30   IX(J)=IX(J)+1
ISN 0045          NR1=NR+1
ISN 0046          DO 40 I=1,NR1
ISN 0047          J=I-1
ISN 0048          IF(IX(I).NE.0)PRINT 300,J,IX(I)
ISN 0050     40   CONTINUE
ISN 0051    300   FORMAT(11XI6,16XI9)
ISN 0052          CALL PET(2)
            C
            C     REARRANGE NUMBERS IN DESCENDING ORDER OF POPULATIONS.
            C     LEAVE 0 AND 1 UNCHANGED SINCE THEY CORRESPOND TO EXTERIOR AND
            C     BOUNDARY POINTS RESPECTIVELY.
            C
ISN 0053          CALL SEQLS(IX(3),IX(NR1+1),NR-1,NR-1)
ISN 0054          PRINT 400
ISN 0055    400   FORMAT('1 REGIONS AFTER REASSIGNMENTS:')
ISN 0056          PRINT 200
ISN 0057          DO 50 I=1,NR1
ISN 0058          J=I-1
ISN 0059          IF(I.LE.2)PRINT 300,J,IX(I)
ISN 0061          IF(I.GT.2)PRINT 350,J,IX(I),IX(I+NR-1)
ISN 0063     50   CONTINUE
ISN 0064    350   FORMAT(11XI6,16XI9,17XI6)
```

```
      [illegible]
      [illegible]
      DO 60 I=3,NR1
   60 IX(IX(I+NR-1)+2)=I-1
      CALL PET(2)
C
C     MODIFY REGION NUMBERS ACCORDING TO NEW ASSIGNMENTS FOUND IN IX.
C
      DO 70 I=1,NREC
      CALL DARN(90,I,IW1,NEL*2)
      DO 80 IEL=1,NEL
      J=IW1(IEL)+1
   80 IW1(IEL)=IX(J)
   70 CALL SAWN(12,IW1,NEL*2)
      CALL PET(2)
      STOP
      END
```

```
      COMPUTER SCIENCES CORPORATION, MAR. 12, 1976.

      SUBROUTINE RIDEN(NTAPB,NREC,NEL,IDEVO,ICMN,IBDY,IW1,IW2,ITABL,
     *     IDENT,MR,NR,LW,MSEG,IRES,IS)
C
C     TO IDENTIFY ALL DISTINCT CONNECTED REGIONS IN A PICTURE SEPARATED
C     BY BOUNDARY LINES.  THE BOUNDARY DATA ARE GIVEN AS NREC RECORDS ON
C     SEQUENTIAL FILE NTAPB, EACH RECORD BEING WRITTEN AS
C     N,(IBDY(I),I=1,N)
C     THE OUTPUT OF THE PROGRAM IS AN NREC*NEL DIRECT ACCESS FILE ON
C     IDEVO CONSISTING OF 0' S FOR BOUNDARY POINTS AND DISTINCT REGION
C     NUMBERS FOR EACH OF THE CONNECTED REGIONS.  ICMN= MINIMUMCOLUMN
C     NUMBER WHICH, ON THE OUTPUT FILE, WILL CORRESPOND TO THE FIRST
C     COLUMN.  IT IS NECESSARY THAT
C        ICMN.LE.MIN(IBDY).LE.MAX(IBDY).LE.ICMN+NEL-1.
C     DEFINE FILE IDEVO(NREC,NEL*2,L,IAV)
C
      DIMENSION IBDY(1)
      LOGICAL*1 IDENT(MR,MR)
      LOGICAL * 1 LW(MR,MR)
      INTEGER*2 IW1(NEL),IW2(NEL),ITABL(MR,MSEG),IS(1)
      DIMENSION IRES(MSEG)
C
C     INITIALIZE A WORK ARRAY IW1 WITH 1'S AND IDENT WITH .FALSE.
C
      CALL SVSCI2(IW1,NEL,1)
      CALL SVSCL1(IDENT,MR*MR,.FALSE.)
      ISEG=1
      IRES(1)=1
      NR=1
C
C     LOOP ON RECORDS
C
      DO 10 IREC=1,NREC
C
C     READ ONE RECORD OF BOUNDARY INFO.
C
      READ(NTAPB)N,(IBDY(I),I=1,N)
      IF(N.NE.0)CALL SORT(IBDY,1,N,N,1,F,TT)
C
C     USE IW1 AND IBDY TO SET ARRAY IW2 AND MATRIX IDENT
C
   30 CONTINUE
      CALL RIDER1 [illegible]1,IBDY,ICMN,NEL,N,IW2,IDENT,MR,NR)
      IF(NR.GT.0)[illegible]TO 25
      PRINT 200,  IREC,NR
  200 FORMAT(' (IREC, NR) = '2I6)
      IF(IREC.EQ.IRES(ISEG))GO TO 40
      NR=-NR
      CALL RIDER4(IDENT,LW,MR,NR,ITABL,ISEG,IW1,NEL,IS,.FALSE.)
      ISEG=ISEG+1
      IF(ISEG.GT.MSEG)GO TO 40
      IRES(ISEG)=IREC
```

```
      IF(NR.LE.MR)GO TO 30
40    PRINT 100,IREC
      NR=0
100   FORMAT(' ERROR CONDITION IN RIDER.  SUPPLIED MR OR MSEG WAS EXCEED
     .ED AT RECORD NUMBER'I6,'.  RETURNING WITH NR=0')
      RETURN
20    CALL DAWN(IDEVO,IREC,IW2,NEL*2)
      CALL VMOVE2(IW2,NEL,IW1)
10    CONTINUE
      CALL RIDER4(IDENT,LW,MR,NR,ITABL,ISEG,IW1,NEL,IS, .TRUE.)
      CALL RIDER2(ITABL,MR*ISEG)
      IRES(ISEG+1)=NREC+1
      PRINT 300
300   FORMAT(///' FINAL TABLES FOR MODIFYING REGION NUMBERS')
      DO 60 JSEG=1,ISEG
      PRINT 400,JSEG
400   FORMAT('0SEGMENT NUMBER'I3)
60    CALL PRTVE2(ITABL(1,JSEG),NR)
      JSEG=1
      DO 70 IREC=1,NREC
      IF(IREC.LT.IRES(JSEG+1))GO TO 50
      JSEG=JSEG+1
50    CONTINUE
      CALL DAWN(IDEVO,IREC,IW1,NEL*2)
      DO 80 IEL=1,NEL
      I=IW1(IEL)
      IF(I.NE.0)IW1(IEL)=ITABL(I,JSEG)
80    CONTINUE
70    CALL DAWN(IDEVO,IREC,IW1,NEL*2)
      NR=0
      CALL VMAXI2(ITABL,MR*ISEG,NR)
      RETURN
      END
```

```
      SUBROUTINE PET(I)
      IF(I.NE.0)GO TO 10
      CALL TIMER(ITIME1)
      TTIME=0.
      WRITE(6,200)
200   FORMAT(10X,'BEGINNING TIMING***   TIME NOW IS 0')
      RETURN
10    CALL TIMER(ITIME2)
      TIME=(ITIME2-ITIME1)/100.
      TTIME=TTIME+TIME
      ITIME1=ITIME2
      WRITE(6,100)TIME,TTIME
100   FORMAT(10X'TIME ELAPSED SINCE LAST PRINTING OF TIME='E12.3,
     .       'SEC.,  TOTAL TIME ELAPSED='E12.3,'SEC.')
      RETURN
      END
```

```
      SUBROUTINE SVSCI2(IX,N,IS)
      INTEGER*2 IX(N)
      DO 10 I=1,N
10    IX(I)=IS
      RETURN
      END
```

```
ISN 0002           SUBROUTINE SORT(A,II,JJ,MM,NN,T,TT)
ISN 0003           DIMENSION A(MM,NN),T(NN),TT(NN),IU(16),IL(16)
ISN 0004           INTEGER A,T,TT
ISN 0005           M=1
ISN 0006           I=II
ISN 0007           J=JJ
ISN 0008      5    IF(I.GE.J)GO TO 70
ISN 0010     10    K=I
ISN 0011           IJ=(I+J)/2
ISN 0012           CALL MVMAMR(A,MM,NN,T,1,IJ,1)
ISN 0013           IF(A(I,1).LE.T(1))GO TO 20
ISN 0015           CALL MVMAMR(A,MM,NN,A,MM,I,IJ)
ISN 0016           CALL MVMAMR(T,1,NN,A,MM,1,I)
ISN 0017           CALL MVMAMR(A,MM,NN,T,1,IJ,1)
ISN 0018     20    L=J
ISN 0019           IF(A(J,1).GE.T(1))GO TO 40
ISN 0021           CALL MVMAMR(A,MM,NN,A,MM,J,IJ)
ISN 0022           CALL MVMAMR(T,1,NN,A,MM,1,J)
ISN 0023           CALL MVMAMR(A,MM,NN,T,1,IJ,1)
ISN 0024           IF(A(I,1).LE.T(1))GO TO 40
ISN 0026           CALL MVMAMR(A,MM,NN,A,MM,I,IJ)
ISN 0027           CALL MVMAMR(T,1,NN,A,MM,1,I)
ISN 0028           CALL MVMAMR(A,MM,NN,T,1,IJ,1)
ISN 0029           GO TO 40
ISN 0030     30    CALL MVMAMR(A,MM,NN,A,MM,K,L)
ISN 0031           CALL MVMAMR(TT,1,NN,A,MM,1,K)
ISN 0032     40    L=L-1
ISN 0033           IF(A(L,1).GT.T(1))GO TO 40
ISN 0035           CALL MVMAMR(A,MM,NN,TT,1,L,1)
ISN 0036     50    K=K+1
ISN 0037           IF(A(K,1).LT.T(1))GO TO 50
ISN 0039           IF(K.LE.L)GO TO 30
ISN 0041           IF(L-I.LE.J-K)GO TO 60
ISN 0043           IL(M)=I
ISN 0044           IU(M)=L
ISN 0045           I=K
ISN 0046           M=M+1
ISN 0047           GO TO 80
ISN 0048     60    IL(M)=K
ISN 0049           IU(M)=J
ISN 0050           J= L
ISN 0051           M=M+1
ISN 0052           GO TO 80
ISN 0053     70    M=M-1
ISN 0054           IF(M.EQ.0)RETURN
ISN 0056           I=IL(M)
ISN 0057           J=IU(M)
ISN 0058     80    IF(J-I.GE.11)GO TO 10
ISN 0060           IF(I.EQ.II)GO TO 5
ISN 0062           I=I-1
ISN 0063     90    I=I+1
ISN 0064           IF(I.EQ.J)GO TO 70
ISN 0066           CALL MVMAMR(A,MM,NN,T,1,I+1,1)
ISN 0067           IF(A(I,1).LE.T(1))GO TO 90
ISN 0069           K=I
ISN 0070    100    CALL MVMAMR(A,MM,NN,A,MM,K,K+1)
ISN 0071           K=K-1
ISN 0072           IF(T(1).LT.A(K,1))GO TO 100
ISN 0074           CALL MVMAMR(T,1,NN,A,MM,1,K+1)
ISN 0075           GO TO 90
ISN 0076           END
```

```
      SUBROUTINE RIDER1(IX,IBDY,ICHN,NEL,N,IY,IDENT,MR,NR)
C
C     GIVEN CURRENT SET OF BOUNDARY ADDRESSES (IBDY(I),I=1,N) AND THE
C     LAST ARRAY IX, FIND CURRENT ARRAY IY CONTAINING REGION IDENTIFICA-
C     TION NUMBERS.  ALSO, IF THE NONBOUNDARY ELEMENTS IN CURRENT ROW
C     ARE CONTIGUOUS WITH ANY NONBOUNDARY POINTS OF THE LAST ROW, SET
C     THE CORRESPONDING ELEMENTS IN IDENT MATRIX.
C
      DIMENSION IBDY(N)
      INTEGER*2 IX(NEL),IY(NEL)
      LOGICAL*1 IDENT(MR,MR)
      IF(N.NE.0)GO TO 10
      CALL SVSCI2(IY,NEL,1)
      GO TO 20
10    CONTINUE
      NRP=NR
C
C     ALL POINTS TO THE LEFT OF IBDY(1) ARE 'EXTERIOR' POINTS (REGION 1)
C
      I=IBDY(1)-ICHN
      IF(I.GT.0)CALL SVSCI2(IY,I,1)
C
C     ALL POINTS TO THE RIGHT OF IBDY(N) ARE 'EXTERIOR' POINTS.
C
      I=IBDY(N)-ICHN+2
      IF(NEL.GE.I)CALL SVSCI2(IY(I),NEL-I+1,1)
C
C     DESIGNATE ALL BOUNDARY POINTS AS 'REGION 0'.
C
      DO 30 I=1,N
      J=IBDY(I)-ICHN+1
30    IY(J)=0
C
C     FOR I=1,N-1 EXAMINE IX(J) FOR IBDY(I).LT.J.LT.IBDY(I+1)
C     AND SET IY ACCORDINGLY.  A NEW REGION NUMBER IS STARTED WHEN IX(J)
C     IS 0 FOR ALL J IN THE ABOVE RANGE.
C
      IF(N.EQ.1)GO TO 20
      N1=N-1
      DO 40 I=1,N1
      L1=IBDY(I)-ICHN+2
      L2=IBDY(I+1)-ICHN
      IF(L1.GT.L2)GO TO 40
      DO 50 L=L1,L2
      IF(IX(L).EQ.0)GO TO 50
      LL=L
      GO TO 60
50    CONTINUE
      NR=NR+1
      IF(NR.LE.MR)GO TO 70
      NR=-NRP
      RETURN
70    CALL SVSCI2(IY(L1),L2-L1+1,NR)
      IDENT(NR,NR)=.TRUE.
      GO TO 40
60    LL=IX(LL)
      CALL SVSCI2(IY(L1),L2-L1+1,LL)
40    CONTINUE
20    CONTINUE
C
C     SET IDENT MATRIX TO INDICATE REGION NUMBERS CORRESPONDING TO
C     IDENTICAL REGIONS.
C
      DO 80 IEL=1,NEL
      I=IX(IEL)
      IF(I.EQ.0)GO TO 80
      J=IY(IEL)
      IF(J.EQ.0)GO TO 80
      IDENT(I,J)=.TRUE.
80    CONTINUE
      RETURN
      END
```

```
      COMPUTER SCIENCES CORPORATION, MAR. 12, 1976.
      SUBROUTINE WIDER2(ITABL,NR)
C
C     CHANGE REGION NUMBERS SUCH THAT CONSECUTIVE NUMBERS ARE USED.
C
      INTEGER*2 ITABL(NR,2)
C
C     FIND THE SET OF NUMBERS IN ITABL(*,1).
C
      DO 5 I=1,NR
    5 ITABL(I,2)=0
      DO 10 I=1,NR
      J=ITABL(I,1)
   10 IF(J.NE.0)ITABL(J,2)=ITABL(J,2)+1
      PRINT 100
      CALL PRTVE2(ITABL(1,2),NR)
C
C     CHANGE ITABL(*,2) TO GET A LOOKUP TABLE FOR ITABL(*,1).
C
      J=0
      DO 20 I=1,NR
      IF(ITABL(I,2).EQ.0)GO TO 20
      J=J+1
      ITABL(I,2)=J
   20 CONTINUE
      PRINT 200
      CALL PRTVE2(ITABL(1,2),NR)
C
C     CHANGE ITABL(*,1).
C
      DO 30 I=1,NR
   30 ITABL(I,1)=ITABL(ITABL(I,1),2)
      RETURN
  100 FORMAT('ONUMBERS OF OCCURENCES OF REGION NUMBERS IN TH ABOVE TABLE
     .(S)')
  200 FORMAT('OLOOK-UP TABLE  TO CHANGE NUMBERS IN THE ABOVE TABLE(S)')
      END
```

```
ISN 0002          SUBROUTINE VMAXI4(IXI4,N,MAXI4)
ISN 0003          DIMENSION IXI4(N)
ISN 0004          DO 10 I=1,N
ISN 0005      10  MAXI4=MAX0(IXI4(I),MAXI4)
ISN 0006          RETURN
            C
ISN 0007          ENTRY VMINI4(IXI4,N,MINI4)
ISN 0008          DO 20 I=1,N
ISN 0009      20  MINI4=MIN0(IXI4(I),MINI4)
ISN 0010          RETURN
ISN 0011          END
```

```
ISN 0002          SUBROUTINE SVSCL1(IX,N,L)
ISN 0003          LOGICAL * 1 IX(N),L
ISN 0004          DO 10 I=1,N
ISN 0005      10  IX(I)=L
ISN 0006          RETURN
ISN 0007          END
```

```
      SUBROUTINE RIDER4(IDENT,LW,MR,NR,ITABL,ISEG,IW1,NEL,IS,LAST)
      LOGICAL LAST
      LOGICAL*1 IDENT(MR,MR)
      INTEGER*2 ITABL(MR,1),LW(MR),IS(1),IW1(NEL)
C
C     D'N ITABL(MR,ISEG),IS(MCR) WHERE MSEG IS THE MAXIMUM
C     NUMBER OF SEGMENTS EXPECTED FOR HANDLING THE GIVEN BOUNDARY IMAGE
C     MCR=MAX NUMBER OF REGION NUMBERS EXPECTED TO OCCUR IN ANY RECORD.
C
C     THIS ROUTINE IS CALLED FROM RIDER WHEN ALL RECORDS ARE PROCESSED
C     (LAST=.TRUE.) OR WHEN THE NUMBER OF REGION NUMBERS FOUND WHILE TEST-
C     ING (IREC+1)'TH RECORD EXCEEDS MR. (LAST=.FALSE.).
C     THEN,
C         1. THE REGION CONNECTIVITY MATRIX IDENT IS REDUCED TO GET A LOOK-
C     UP TABLE FOR THE CURRENT SEGMENT.
C         2. THE LOOK-UP TABLES CORRESPONDING TO EARLIER SEGMENTS ARE
C     MODIFIED BASED ON NEWLY FOUND CONNECTIVITIES, IF ANY.
C        3. THE DISTINCT REGION NUMBERS OCCURING IN THE IREC'TH RECORD
C     ARE FOUND; A CORRESPONDENCE ARRAY IS BETWEEN CURRENT AND NEXT SEGMENT
C     SET UP.  THE LAST RECORD(IW1) IS MODIFIED TO MATCH THE NUMBERING
C     THE NEXT SEGMENT.
C        4. THE CONNECTIVITY MATRIX IS MODIFIED TO PRESERVE THE INFORMA-
C     TION ON THE CONNECTIONS BETWEEN REGIONS IN IREC'TH RECORD.
C
C                 SECTION 1.
C
      DO 50 I=1,NR
      DO 50 J=1,NR
50    IDENT(I,J)=IDENT(I,J).OR.IDENT(J,I)
      CALL RIDER5(IDENT,MR,NR,ITABL(1,ISEG),LW)
      ISEGT=(ISEG-1)*MR
      DO 10 I=1,NR
10    IF(ITABL(I,ISEG).GT.1)ITABL(I,ISEG)=ITABL(I,ISEG)+ISEGT
      IF(MR.GT.NR)CALL SVSCI2(ITABL(NR+1,ISEG),MR-NR,0)
      PRINT 100,ISEG
      CALL PRTVE2(ITABL(1,ISEG),NR)
C
C               SECTION 2.
C
      IF(ISEG.EQ.1)GO TO 60
      ISEG1=ISEG-1
      CALL RIDER7(ITABL(1,ISEG),IS,NCR,ITABL,MR*ISEG1)
      DO 40 JSEG=1,ISEG1
      KSEG=ISEG-JSEG
      PRINT 200,KSEG
40    CALL PRTVE2(ITABL(1,KSEG),MR)
C
C               SECTION 3.
C
60    IF(LAST)RETURN
      CALL RIDER6(IW1,NEL,IS,NR)
      NCR=NR
      PRINT 300,NR
      CALL PRTVE2(IS,NR)
      DO 70 I=1,NR
70    IS(I)=ITABL(IS(I),ISEG)
      PRINT 400
      CALL PRTVE2(IS,NR)
```

```
C             SECTION 4.
C     CONNECTIVITIES BETWEEN NEW REGIONS I,J IN THE LAST RECORD ARE FOUND
C     BY TESTING WHETHER IS(I).EQ.IS(J)
C
      CALL SVSCL1(IDENT,NR*NR,.FALSE.)
      DO 20 I=1,NR
      IDENT(I,I)=.TRUE.
      IF(I.EQ.NR)GO TO 20
      I1=I+1
      DO 30 J=I1,NR
30    IDENT(I,J)=IS(I).EQ.IS(J)
20    CONTINUE
      RETURN
100   FORMAT(///' THE FOLLOWING IS A PRELIMINARY TABLE FOR MODIFYING SEG
     .MENT NUMBER'I3)
200   FORMAT('0THE FOLLOWING IS AN UPDATED TABLE FOR MODIFYING SEGMENT N
     .UMBER'I3)
300   FORMAT(' THE DISTINCT REGION NUMBERS PRESENT IN THE LAST RECORD OF
     . THE CURRENT SEGMENT(TO BE REASSIGNED NOS. 1 THROUGH'I4/' IN THE N
     .EXT SEGMENT):')
400   FORMAT(' ASSIGNMENTS FOR THE ABOVE REGIONS FROM THE PRELIMINARY LO
     .OK-UP TABLE:')
      END
```

COMPUTER SCIENCES CORPORATION, MAR. 12, 1976.

```
      SUBROUTINE RIDERS(IDENT,ND,N,IT,M)
C
C     TO GENERATE A TABLE IT MAPPING J=1,...,N TO I=IT(J)= SMALLEST K
C     SUCH THAT THERE EXISTS A SEQUENCE (K(ID),ID=1,...,L) WITH K(1)=I,
C     K(L)=J AND IDENT(K(ID),K(ID+1))=.TRUE.
      INTEGER*2 IT(N),M(1)
      LOGICAL*1 IDENT(ND,N)
      DO 100 I=1,N
100   IT(I)=I
      I=0
10    I=I+1
      IF(I.LE.N)GO TO 20
      RETURN
20    IF(IT(I).LT.I)GO TO 10
      J=0
      K=0
30    J=J+1
      IF(J.LE.N)GO TO 40
      L=0
50    L=L+1
C
C
      IF(L.GT.K)GO TO 10
      J=0
70    J=J+1
      IF(J.GT.N)GO TO 50
      IF(.NOT.IDENT(M(L),J))GO TO 70
      IF(IT(J).EQ.I)GO TO 70
      IT(J)=I
      K=K+1
      M(K)=J
      GO TO 70
40    IF(.NOT.IDENT(I,J))GO TO 30
      IT(J)=I
      K=K+1
      M(K)=J
      GO TO 30
      END
```

COMPUTER SCIENCES CORPORATION, MAR. 12, 1976.

```
      SUBROUTINE RIDER6(IX,M,IS,N)
C
C     FIND A SET IS OF DISTINCT NONZERO ELEMENTS IN IX.  THE NUMBER OF
C     SUCH ELEMENTS IS N.
C
      INTEGER*2 IX(M),IS(1)
      N=1
      IS(1)=1
      DO 10 I=1,M
      IF(IX(I).LE.1)GO TO 10
      IF(N.EQ.0)GO TO 20
      DO 30 J=2,N
   30 IF(IS(J).EQ.IX(I))GO TO 40
   20 N=N+1
      IS(N)=IX(I)
      IX(I)=N
      GO TO 10
   40 IX(I)=J
   10 CONTINUE
      RETURN
      END
```

COMPUTER SCIENCES CORPORATION, MAR. 12, 1976.

```
      SUBROUTINE RIDER7(IX,IS,N,IY,M)
      INTEGER*2 IX(N),IS(N),IY(M)
C
C     MODIFY RELEVANT ENTRIES IN IY ACCORDING TO CONNECTIVITIES FOUND
C     IN IS.
      IF(N.EQ.1)RETURN
      MAX=IS(2)
      MIN=IS(2)
      CALL VMAXI2(IS(2),N-1,MAX)
      CALL VMINI2(IS(2),N-1,MIN)
      DO 10 J=1,M
      IF(IY(J).LT.MIN.OR.IY(J).GT.MAX)GO TO 10
      DO 20 I=2,N
      IF(IY(J).NE.IS(I))GO TO 20
      IY(J)=IX(I)
      GO TO 10
   20 CONTINUE
   10 CONTINUE
      RETURN
      END
```

```
ISN 0002          SUBROUTINE VMOV2(IX,N,IY)
ISN 0003          INTEGER*2 IX(N),IY(N)
ISN 0004          DO 10 I=1,N
ISN 0005     10   IY(I)=IX(I)
ISN 0006          RETURN
ISN 0007          END
```

```
          SUBROUTINE PRTVE2(IX,N)
          INTEGER*2 IX(N)
          PRINT 100,IX
     100  FORMAT(1X25I5)
          RETURN
          END


          SUBROUTINE SEQSL(X,ISEQ,N,ND)
          DIMENSION X(ND,2),ISEQ(ND),T(2),TT(2)
     C
     C    MUST EQUIVALENCE (X(1,2),ISEQ(1))
     C
          IFLAG=0
          GO TO 10
          ENTRY SEQLS(X,ISEQ,N,ND)
          IFLAG=1
      10  DO 20 I=1,N
      20  ISEQ(I)=I
          CALL SORT(X,1,N,ND,2,T,TT)
          IF(IFLAG.EQ.0)RETURN
          CALL FLIPV(X,N,X)
          CALL FLIPV(ISEQ,N,ISEQ)
          RETURN
          END


          SUBROUTINE FLIPV(X,N,Y)
          DIMENSION X(N),Y(N)
     C
     C    ... EQ'CE(X,Y)
     C
          N2=N/2
          N1=N+1
          DO 10 I=1,N2
          W=X(I)
          II=N1-I
          Y(I)=X(II)
          Y(II)=W
      10  CONTINUE
          RETURN
          END


ISN 0002          SUBROUTINE SVSCI(IX,N,IS)
ISN 0003          DIMENSION IX(N)
ISN 0004          DO 10 I=1,N
ISN 0005     10   IX(I)=IS
ISN 0006          RETURN
ISN 0007          END


ISN 0002          SUBROUTINE VMAXI2(IXI2,N,MAXI2)
ISN 0003          INTEGER*2 IXI2(N)
ISN 0004          DO 10 I=1,N
ISN 0005     10   IF(IXI2(I).GT.MAXI2)MAXI2=IXI2(I)
ISN 0007          RETURN
ISN 0008          ENTRY VMINI2(IXI2,N,MINI2)
ISN 0009          DO 20 I=1,N
ISN 0010     20   IF(IXI2(I).LT.MINI2)MINI2=IXI2(I)
ISN 0012          RETURN
ISN 0013          END
```

5-5-5 DELETION OF BOUNDARY POINTS

1 NAME

DBOUND

2 PURPOSE

To modify each of the "0" pixels in an image to the most frequently occurring number in its 3 by 3 neighborhood. (This is useful, for example, in generating a level I GTM from a level II map and/or suppressing all the boundary points in a GTM and replacing them with reasonable class labels).

3 CALLING SEQUENCE

CALL DBOUND (NREC, NEL, NEL2, NTAPI, NTAPO, IX, IY)

where

NREC = Number of records in the input image;
NEL = Number of pixels per record;
NEL2 = NEL+2;
NTAPI, NTAPO are the logical unit numbers of input and output sequential data sets;
IX, IY are work arrays to be dimensioned as indicated in the listings.

All the calling arguments except IX and IY are inputs.

4 INPUT-OUTPUT

Both the input and output sequential data sets have the same format. The number of records is NREC. The number of pixels per record is NEL and the number of bytes per pixel is 4. The records are in unformatted FORTRAN.

5 EXITS

No nonstandard exits

6 USAGE

The program is in FORTRAN IV and implemented on the IBM 360 using the H compiler. The program is in the users' library as a load module.

7 EXTERNAL INTERFACES

The subprograms required by this routine are:

SARN, a sequential access array read routine;
VMOV, a routine to move a vector in core;
MAJOR, a function giving the most frequently occuring number in a 3 by 3 neighborhood.

8 PERFORMANCE SPECIFICATIONS

8.1 Storage

This subroutine is 1036 bytes long. With the main program needed to call it for an image with NEL = 866, the external references and buffers, the storage required is 40K bytes.

8.2 Execution Time

Depends on image size. For a test case of 1624 by 866 pixels it took approximately 100 seconds.

8.3 I/O Load

None

8.4 Restrictions

None

9 METHOD

This program uses a circular buffer IX with pointers I1, I2, I3 indicating the previous, present and next records under consideration. Initially, I1, I2, I3 are set at 1, 2 and 3 respectively. After each record is processed, the pointers I1, I2, I3 are "rolled" upward. The processing of each record consists of checking the eight neighbors of each pixel whose value is zero. The function subprogram MAJOR is employed to determine the most frequent number occuring in the set of eight (If such a number is not unique, the first encountered number is taken).

Records 0 and NREC+1 are defined to be identical to records 1 and NREC respectively. Also, pixels 0 and NEL+1 in any record are defined to be the same as pixels 1 and NEL in the same record.

10 COMMENTS

None

11 LISTINGS

The listings of DBOUND and MAJOR are attached at the end of this section.

12 TESTS

This program was used in removing the extraneous boundary points after conversion of the level II GTM of the Mobile Bay region to a level I map. Line-printer plots of the maps before and after the application of DBOUND indicate satisfactory operation of this program.

```
ISN 0002            SUBROUTINE DBOUND(NREC,NEL,NEL2,NTAPI,NTAPO,IX,IY)
ISN 0003            DIMENSION IX(NEL2,3),IY(NEL)
ISN 0004            NEL4=NEL*4
               C    INITIALIZE ARRAY IX.
               C    THE PURPOSE OF THIS PROGRAM IS TO MODIFY POINTS IN THE IMAGE ON NTAPI
               C    WITH NUMBER 0 TO THE NUMBER OCCURING MOST FREQUENTLY IN A 3 BY 3
               C    NEIGHBORHOOD.
ISN 0005            I1=1
ISN 0006            I2=2
ISN 0007            I3=3
ISN 0008            CALL SARN(NTAPI,IX(2,I1),NEL4)
ISN 0009            IX(1,I1)=IX(2,I1)
ISN 0010            IX(NEL2,I1)=IX(NEL+1,I1)
ISN 0011            CALL VMOV(IX(1,I1),NEL2,IX(2,I2))
ISN 0012            DO 10 I=1,NREC
               C
               C    IF(I.LT.NREC) READ (I+1)'ST RECORD INTO IX(*,I3).
ISN 0013            IF(I.LT.NREC)CALL SARN(NTAPI,IX(2,I3),NEL4)
ISN 0015            IF(I.EQ.NREC)CALL VMOV(IX(2,I2),NEL,IX(2,I3))
ISN 0017            IX(1,I3)=IX(2,I3)
ISN 0018            IX(NEL2,I3)=IX(NEL+1,I3)
               C
               C    NOW, THE PREVIOUS, CURRENT AND NEXT ROWS ARE IN IX(*,I1),IX(*,I2)
               C    AND IX(*,I3) RESPECTIVELY.  MODIFY EACH 0 IN IX(*,I2) TO THE MAJO-
               C    RITY CLASS NUMBER IN THE 3 BY 3 NEIGHBORHOOD OF IT.
ISN 0019            DO 20 J=1,NEL
ISN 0020            IY(J)=IX(J+1,I2)
ISN 0021       20   IF(IY(J).EQ.0)IY(J)=MAJOR(IX,NEL2,I1,I2,I3,J+1)
ISN 0023            WRITE(NTAPO)IY
               C
               C    MODIFY I1,I2,I3 IN PREPARATION FOR THE NEXT RECORD.
ISN 0024            IW=I1
ISN 0025            I1=I2
ISN 0026            I2=I3
ISN 0027            I3=IW
ISN 0028       10   CONTINUE
ISN 0029            RETURN
ISN 0030            END

ISN 0002            FUNCTION MAJOR(IX,NEL,I1,I2,I3,J)
ISN 0003            DIMENSION IX(NEL,3),LABEL(9),NUMBER(9)
               C
               C    FIND THE MOST FREQUENTLY OCCURING NUMBER AMONG THE EIGHT NEIGHBORS
               C    OF IX(J,I2).  NOTE THAT 1.LT.J.LT.NEL.
ISN 0004            LABEL(1)=IX(J-1,I1)
ISN 0005            NUMBER(1)=1
ISN 0006            N=1
ISN 0007            J2=J-2
ISN 0008            DO 30 I=1,3
ISN 0009            IF(I.EQ.1)II=I1
ISN 0011            IF(I.EQ.2)II=I2
ISN 0013            IF(I.EQ.3)II=I3
ISN 0015            KM=1
ISN 0016            IF(I.EQ.1)KM=2
ISN 0018            INC=1
ISN 0019            IF(I.EQ.2)INC=2
ISN 0021            DO 10 K=KM,3,INC
ISN 0022            DO 20 L=1,N
ISN 0023       20   IF(IX(J2+K,II).EQ.LABEL(L))GO TO 40
ISN 0025            N=N+1
ISN 0026            LABEL(N)=IX(J2+K,II)
ISN 0027            NUMBER(N)=1
ISN 0028            GO TO 10
ISN 0029       40   NUMBER(L)=NUMBER(L)+1
ISN 0030       10   CONTINUE
ISN 0031       30   CONTINUE
ISN 0032            MAX=0
ISN 0033            DO 50 I=1,N
ISN 0034            IF(NUMBER(I).LE.MAX)GO TO 50
ISN 0036            MAJOR=LABEL(I)
ISN 0037            MAX=NUMBER(I)
ISN 0038       50   CONTINUE
ISN 0039            RETURN
ISN 0040            END
```

5-5-6 THICKENING OF DIGITALLY DEFINED CURVES

1 NAME: THICK2

2 PURPOSE: To modify curve information in scan line intersection code so as to represent two-dimensionally thickened curves.

3 CALLING SEQUENCE:

CALL THICK2(NTAPI,NTAPO,IX,IY,IW,NREC,K)

where

NTAPI = logical unit number of input sequential data set.

NTAPO = logical unit number of output sequential data set.

IX, IY, IW are work arrays.

IX and IY should be dimensioned N where N = Maximum number of intersections of the thickened curve with (2K+1) successive scan lines (see Section 9).

IW should be dimensioned (2K+1).

NREC = Number of records in the input (or output) image.

K = Number of elements by which the image should be thickened.

NTAPI, NTAPO, NREC and K are inputs to this routine.

4 INPUT-OUTPUT

The input to this program is a curve stored in SLIC format on unit NTAPI. NREC records are stored as J, (IX(L), L = 1,J) in FORTRAN binary format where J = number of coordinates in the record and IX is the array of coordinates.

The output of this program will consist of NREC records on unit NTAPO in the same format as of the input.

5 EXITS: No non-standard exits.

6 USAGE: The program exists in both IBM-7094 and IBM-360 versions and is written in FORTRAN IV. The decks are available with the author.

7 EXTERNAL INTERFACES:

7.1 System Routines: IBCOM#

7.2 Other Programs Called: SVSCI, SORT, ELIRPT, THICK1, VMOV

7.3 External Storage: None.

8 PERFORMANCE SPECIFICATIONS:

8.1 Storage: 518 hexadecimal bytes. Including the routines named in Section 7.2, the storage required is 14C2 hexadecimal bytes. (This does not include the storage needed for the work arrays which is data dependent.)

8.2 Execution Time: Depends largely on the number of points on the curve to be thickened and K. A test run on a file with 1753 records and approximately 12000 points on the curve took 6.8 minutes with K=2 on the IBM 360/65 system.

8.3 Restrictions: None.

9 METHOD:

The routine essentially consists of thickening in the horizontal direction by calls to THICK1 and taking unions of 2K+1 successive records to achieve thickening in the vertical direction.

The array IW is used to store the number of coordinates after thickening in the horizontal direction. IW(1) through IW(2K+1) are the numbers of coordinates in the (2K+1) successive records which are combined to form the current record of output.

Initially, the components of IW are all set to 0. Next (K+1) records of input are read, thickened in the horizontal direction and all the resulting boundary coordinates are stored in array IY. After the I'th record is thickened, the number of coordinates corresponding to it is stored in IW(I+K).

Now, the following operations are performed for I+1 through NREC. The coordinates in IY are moved into IX. IX is sorted, repetitions are eliminated and an output record is written. Next, the array IY is left-shifted by IW(1) words since the first IW(1) words (which correspond to the earliest horizontally thickened input record) are no longer required. Next, IW is left-shifted by one word. Now, if there are any more input records left, the next input record is read into IX and thickened. The thickened coordinate values are always loaded into the right end of IY and the number of coordinates is stored in IW(2K+1).

The routine THICK1 operates as follows. It assumes that the input array IX is in ascending order. First, it sets IY(1) through IY(2K+1) to IX(1)-K through IX(1)+K. This corresponds to thickening the first point in IX. After the I'th point is thickened, suppose n values have been produced in IY. Then, the values corresponding to thickening the $(I+1)^{st}$ point are Max(IX(I+1)-K, IY(n)+1) through IX(I+1)+K. When an M-vector IX is thickened by K elements using THICK1, the number N of components in the output array IY is bounded by

$$2K + M \leq N \leq (2K + 1) M .$$

10 COMMENTS: None.

11 LISTING: A listing of THICK1 and THICK2 is attached at the end of this section.

12 **TESTS:** This program has been tested on synthetic data and on the boundary data for the TARCOG counties in North Alabama and found to work satisfactorily.

```
      SUBROUTINE THICKN(IX,M,IY,N,K)
C     TO THICKEN VECTOR IX BY K ELEMENTS AND PUT IN IY.
      DIMENSION IX(M),IY(1)
C     D'N IY(EXPECTED NUMBER OF VALUES AFTER THICKENING)
      N=0
      IF(M.EQ.0)RETURN
      N=1
      IY(1)=IX(1)-K
      I=1
50    IFIN=IX(I)+K
20    IF(IY(N).EQ.IFIN)GO TO 10
      N=N+1
      IY(N)=IY(N-1)+1
      GO TO 20
10    IF(M.EQ.I)RETURN
40    I=I+1
      IF(IY(N).LT.IX(I)+K.OR.I.EQ.M)GO TO 90
      GO TO 40
90    N=N+1
      IY(N)=MAX0(IX(I)-K,IY(N-1)+1)
      GO TO 90
      END
```

```
      SUBROUTINE THICK2(NTAPI,NTAPO,IX,IY,IW,NREC,K)
      DIMENSION IX(1),IY(1),IW(1)

C     TO THICKEN BOUNDARY INFO ON TAPE NTAPI IN TWO DIMENSIONS BY K
C     ELEMENTS ON EACH SIDE OF TEH BOUNDARY POINTS AND WRITE ON NTAPO.

      K2=K*2
      K21=K2+1
      K1=K+1
      NRECK1=NREC-K1
      K1=K+1
      CALL SVSCI(IW,K21,0)

C     READ K+1 RECORDS OF INPUT, THICKEN IN ONE DIMENSION AND INITIALIZE
C     IY.

      N=1
      DO 10 I=1,K1
      IK=I+K
      READ(NTAPI)J,(IX(L),L=1,J)
      IF(J.EQ.0)GO TO 15
      CALL SORT(IX,1,J ,J ,1,T,TT)
      CALL ELIMPT(J,IX)
   15 CONTINUE
      CALL THICK1(IX,J,IY(N),IW(IK),K)
   10 N=N+IW(IK)
      DO 20 I=1,NREC
      N1=N-1
      IF(N1.EQ.0)GO TO 30
      CALL VMOV(IY,N1,IX)
      CALL SORT(IX,1,N1,N1,1,T,TT)
      CALL ELIMPT(N1,IX)
   30 WRITE(NTAPO)N1,(IX(L),L=1,N1)

C     UPDATE ARRAY IY BY READING NEW RECORD OF INPUT.

      IW1=IW(1)+1
      N1=N-IW1
      IF(IW(1).NE.0.AND.N1.NE.0)CALL VMOV(IY(IW1),N1,IY)
      N=N-IW(1)
      CALL VMOV(IW(2),K2,IW)
      IF(I.GT.NRECK1)GO TO 20
      READ(NTAPI)J,(IX(L),L=1,J)
      IF(J.EQ.0)GO TO 25
      CALL SORT(IX,1,J ,J ,1,T,TT)
      CALL ELIMPT(J,IX)
   25 CONTINUE
      CALL THICK1(IX,J,IY(N),IW(K21),K)
      N=N+IW(K21)
   20 CONTINUE
   99 FORMAT(1X,2I5,25I4)
      RETURN
      END
```

REFERENCES

1. Freeman, H., "Computer Processing of Line Drawing Images," Computing Surveys, March 1974, p. 57.

2. Thomas, V. L., "Generation and Physical Characteristics of the ERTS MSS System Corrected Computer Compatible Tapes," GSFC Report X-563-73-206, July 1973.

3. Nagy, G., "State-of-the-Art in Pattern Recognition," Proc. IEEE, May 1968, p. 836.

4. Nilsson, N. J., Learning Machines, McGraw-Hill, New York, 1965.

5. Ho, Y. C. and Kashyap, R. L., "A Class of Iterative Procedures for Linear Equalities," J. Siam on Control, Vol. 4, 1966, p. 112.

6. Ho, Y. C. and Kashyap, R. L., "An Algorithm for Linear Inequalities and Its Applications," IEEE Trans. Electronic Computers, October 1965, p. 683.

7. Bonner, W. D., "Gridding Scheme for APT Satellite Pictures," J. of Geophysical Research, August 1969, p. 4581.

8. Kratky, V., "Precision Processing of ERTS Imagery," Technical Papers from the 1971 ASP Fall Convention, San Francisco, 1971, p. 481.

9. Landsat Data Users Handbook, NASA/GSFC, April 1977, Update.

10. Kratky, V., "Cartographic Accuracy of ERTS," Photogrammetric Engineering, 1974, p. 203.

11. Barnea, D. S. and Silverman, H. F., "A Class of Algorithms for Fast Digital Image Registration," IEEE Trans. Computers, February 1972.

12. Ramapriyan, H. K., " A Multilevel Approach to Sequential Detection of Pictorial Features," IEEE Trans. Computers, January, 1976.

13. Bernstein, R., "Digital Image Processing of Earth Observation Sensor Data," IBM J. Res. Develop., January, 1976, p. 40.

14. Holz, Robert K., Huff, David L., and Mayfield, Robert C., "Urban Spatial Structure Based on Remote Sensing Imagery," Proc. Sixth International Symposium Remote Sensing Environment, October 1969, p. 819.

Table of Contents

Table of Contents (Cont'd.)

<u>LIST OF ILLUSTRATIONS</u>

<u>Figure</u>

LIST OF ILLUSTRATIONS (Cont'd.)

Figure

LIST OF TABLES

15. Rosenfeld, A., "Connectivity in Digital Pictures," Journal of the ACM, Vol. 20, No. 1, January 1973, pp. 81-87.

16. Merrill, R. D., "Representation of Contours and Regions for Efficient Computer Search," Communications of the ACM, Vol. 16, No. 2, February 1973, pp. 69-82.

17. Downs, Sanford W., Shurma, G. C., and Bagwell, Colin, "A Procedure Used for a Ground Truth Study of a Land Use Map of North Alabama Generated from Landsat Data," NASA TN.

18. Malila, W. A. and Nalepka, R. F., "Advanced Processing and Information Extraction Techniques Applied to ERTS-1 MSS Data." Third Earth Resources Technology Satellite-1 Symposium, Dec. 10-14, 1973,Goddard Space Flight Center, p. 1743.

19. Horowitz, H., et al, "Estimating the Proportions within a Single Resolution Element of a Multispectral Scanner," Proceedings of the 7th International Symposium on Remote Sensing of Environment, May 1971, Ann Arbor, Michigan.

20. M. Lybanon, "Kohoutek Photometry Data Analysis Program Documentation - Partial Draft," CSC Memorandum to File, 5E3010-16, March 29, 1976.

21. Ramapriyan, H. K., "Geometric Correction of Remotely Sensed Images" CSC Memorandum to File, 5E3030-1-1, September 13, 1974.

22. H. K. Ramapriyan, "GEOGREF - A Program for Determining the Parameters for Geographic Referencing", CSC Memorandum to File, 5E3090-1-8, July 30, 1976.

23. H. K. Ramapriyan, "Programs for Digital Manipulation of Curves sucha s Political Boundaries", CSC Memorandum to File, 5E3030-1-2-3, January 24, 1975.

24. A. Rosenfeld, Picture Processing by Computer, Academic Press, 1969.

TECHNICAL REPORT STANDARD TITLE PAGE

1. REPORT NO. NASA CR-2932	2. GOVERNMENT ACCESSION NO.	3. RECIPIENT'S CATALOG NO.
4. TITLE AND SUBTITLE Digital Computer Processing of LANDSAT Data for North Alabama		5. REPORT DATE December 1977
		6. PERFORMING ORGANIZATION CODE
7. AUTHOR(S) A. D. Bond, R. J. Atkinson, M. Lybanon and H. K. Ramapriyan		8. PERFORMING ORGANIZATION REPORT # M-240
9. PERFORMING ORGANIZATION NAME AND ADDRESS Computer Sciences Corporation, Aerospace Systems Center 8300 Whitesburg Drive Huntsville, Alabama 35802		10. WORK UNIT NO.
		11. CONTRACT OR GRANT NO. NAS8-21805
		13. TYPE OF REPORT & PERIOD COVERED Contractor Report
12. SPONSORING AGENCY NAME AND ADDRESS National Aeronautics and Space Administration Washington, D. C. 20546		14. SPONSORING AGENCY CODE
15. SUPPLEMENTARY NOTES		

16. ABSTRACT

Computer processing procedures and programs applied to Multispectral Scanner (MSS data from Landsat are described. The output product produced is a Level I land use map in conformance with a Universal Transverse Mercator (UTM) projection. The region studied was a five-county area in North Alabama.

17. KEY WORDS		18. DISTRIBUTION STATEMENT STAR Category 43	
19. SECURITY CLASSIF. (of this report) Unclassified	20. SECURITY CLASSIF. (of this page) Unclassified	21. NO. OF PAGES 283	22. PRICE $9.25

* For sale by the National Technical Information Service, Springfield, Virginia 22161

END

8.23.78